Fuzzy modelling in reinforcement learning

Quah, Kian Hong

2006

Quah, K. H. (2006). Fuzzy modelling in reinforcement learning. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2428>

<https://doi.org/10.32657/10356/2428>

Nanyang Technological University

Downloaded on 26 Apr 2023 20:33:09 SGT

a407011

Fuzzy Modelling in Reinforcement Learning

Quah Kian Hong

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2006

Q
325.6
Q1
2006

Acknowledgements

I greatly acknowledge the guidance, the advice, the motivation and the support of my Thesis advisor, Associate Professor Quek Hiok Chai, over the past seven years, during my undergraduate studies, my career development in the industry, and my postgraduate research. His profound knowledge in artificial intelligence and control theory has inspired and has shaped my directions into this promising area of research. His criticism and guidance on my writing, organization, and other research skills have been exceptionally constructive towards the success of this research.

I am grateful to A/P Graham Leedham, A/P Ng Geok See, A/P Abdul Wahab, and A/P Man Zhihong from our school for their helpful discussions and assistances in my research.

Interactions have been rewarding with students at the Centre for Computational Intelligence (formerly Intelligent Systems Laboratory) and within the School of Computer Engineering. Especially, I would like to acknowledge Tung Whye Loon, Ting Chan Wai and Ang Kai Keng, with whom I have discussed a number of research ideas and directions that constituted part of this research. Other students who motivate this lively and friendly research experiences are Goh Ghee Ming, Nguyen Viet Thang, Wong Lai Ping, Md. Saiful Islam, Ho Nhu Binh, Ma Li, Gao Shiyu, Ho Sy Loi, Nguyen Ngoc Minh, Lim Teck Meng, Ng Wei Keng, and Ong Te Min. I would also like to thank the laboratory technicians, Tan Swee Huat and Lim Hong Heng, for their helpful supports.

I would like to extend my gratitude to Nanyang Technological University for sponsoring my research scholarship for three years.

Last but not least, I am indebted to my family and relatives, especially to my parents for their encouragement, nurturance and educational supports. Most importantly, I deeply appreciate the love, the companionship and the understanding of my fiancée, Ng Wee Ding.

Contents

Acknowledgements.....	i
Contents	ii
List of Figures.....	iv
List of Tables	v
List of Abbreviations	vi
Summary	vii
1 Introduction.....	1
1.1 Motivations	1
1.2 Objectives	3
1.3 Major Contributions.....	3
1.4 Organization of the Thesis	5
2 Review of Reinforcement Learning and Fuzzy Systems.....	6
2.1 Reinforcement Learning	6
2.1.1 Basic Mathematical Definitions.....	7
2.1.2 Reinforcement Learning Models	10
2.1.3 Function Approximations in Reinforcement Learning.....	17
2.2 Fuzzy Systems	21
2.2.1 Fuzzy Sets, Fuzzy Operations, and Fuzzy Rules	21
2.2.2 Fuzzy Modelling	23
2.2.3 Fuzzy Reinforcement Learning Models.....	27
3 FITSK: The Generic Fuzzy Input Takagi-Sugeno-Kang Fuzzy Framework.....	32
3.1 FITSK	33
3.1.1 Generic Structure of FITSK.....	34
3.1.2 A Zero-Order FITSK Model.....	38
3.1.3 A First-Order FITSK Model.....	43
3.2 Benchmarking Results and Analysis	50
3.2.1 Task (a): Nonlinear system.....	52
3.2.2 Task (b): Human control of chemical plant.....	53
3.2.3 Task (c): Daily price of a stock in a stock market	54
3.2.4 Task (d): Mackey-Glass time series prediction	56
3.2.5 Task (e): Motorcycle crashing problem.....	58
3.3 Summary Remarks.....	61
4 FITSK-RL: Function Approximation with FITSK for a Generic Reinforcement Learning Framework	66
4.1 A Generic Reinforcement Learning Framework	68
4.1.1 Functionality of Modules.....	69
4.1.2 Mapping to Autonomous and Controlled Models	71
4.2 FITSK-RL Model.....	72
4.2.1 The Zero-order FITSK model.....	73
4.2.2 Discrete Action FITSK-RL.....	75
4.2.3 Continuous Action FITSK-RL.....	78
4.3 Simulation Results and Analysis	81
4.3.1 Puddle World task.....	82
4.3.2 Mountain-Car task	85
4.4 Summary Remarks.....	93
5 Maximum Reward Reinforcement Learning: A Non-Cumulative Reward Criterion	95
5.1 Maximum Reward Reinforcement Learning	96
5.1.1 Formulation of Maximum Reward Reinforcement Learning	97
5.1.2 Mapping to a Generic Reinforcement Learning Framework.....	100

5.2	Comparison with Q-learning and R-learning.....	101
5.3	Optimal Stopping Problem in Financial Derivative Pricing	104
5.3.1	Problem Formulation	105
5.3.2	Learning Model Formulation and Results	106
5.3.3	Discussion.....	109
5.4	Summary Remarks.....	109
6	MCES: The Monte Carlo Evaluative Selection Approach	112
6.1	Background Studies	114
6.1.1	Problem definition	114
6.1.2	Regressive Task Feature Selection methods.....	115
6.1.3	Dimensions of Feature Selection methods.....	119
6.2	Monte Carlo Evaluative Selection	123
6.2.1	Assumptions for MCES	124
6.2.2	MCES Algorithm	125
6.2.3	Comparison with Sensitivity-based Pruning.....	129
6.3	Benchmarking Results	130
6.3.1	Rule-plus-exception problem.....	131
6.3.2	Automobile MPG Prediction	131
6.3.3	Audio Detection problem.....	134
6.3.4	Stock market prediction	136
6.4	Summary Remarks.....	138
7	Conclusions and Recommendations	140
7.1	Conclusions.....	140
7.2	Recommendations for Future Research	143
	References.....	145
	Author's Publications.....	158

List of Figures

Figure 2-1: A typical reinforcement learning paradigm.	6
Figure 2-2: A sequence of Markov decision process.	7
Figure 2-3: Approaches to solve Markov decision processes.	10
Figure 2-4: Actor-critic model.	14
Figure 2-5: The 7-star problem for value iteration.	20
Figure 3-1: Generic Structure of FITSK.	34
Figure 3-2: The motorcycle crash dataset.	41
Figure 3-3: Training and testing mean-squared-errors of FITSK.	42
Figure 3-4: Time-complexity of FITSK with Kalman filter learning.	45
Figure 3-5: Time-complexity of FITSK with localized Kalman filter learning.	49
Figure 3-6: Flowchart on 3-criteria model selection of FITSK.	50
Figure 3-7: Testing result of FITSK ⁰ for task (a).	53
Figure 3-8: Testing result of FITSK ⁰ for task (b).	54
Figure 3-9: Testing result of FITSK ⁰ for task (c).	55
Figure 3-10: Global and local learning results of first-order FITSK models.	59
Figure 3-11: Learning errors of FITSK ¹ with different degree of localization.	60
Figure 3-12: Learning results of direct vs. momentum update formulae.	64
Figure 4-1: A generic reinforcement learning framework.	68
Figure 4-2: Aspects of FITSK-RL for benchmarking simulations.	81
Figure 4-3: Number of steps to reach goal for each trial.	83
Figure 4-4: Trajectories of optimal (36 steps) and sub-optimal (46 steps) policies.	83
Figure 4-5: (a) Cost-to-goal, (b) action dynamics; during the 500 th trial of learning.	84
Figure 4-6: Mountain Car task.	86
Figure 4-7: Average number of steps to reach goal for FITSK-RL.	86
Figure 4-8: Average number of steps to reach goal for tile-coding.	87
Figure 4-9: Trajectories of near-optimal and sub-optimal policies.	88
Figure 4-10: Number of steps to reach goal (without shrinking of membership).	88
Figure 4-11: Cost-to-goal function of FITSK-RL.	89
Figure 4-12: (a) Cost-to-goal, (b) action dynamics; during the 20000 th trial.	90
Figure 4-13: Average number of steps to reach goal (FITSK-RL with Q-learning).	91
Figure 4-14: Average number of steps to reach goal (continuous action FITSK-RL).	92
Figure 5-1: An MDP with two cycles.	102
Figure 5-2: Comparison of R-learning, Q-learning and Q _M -learning.	103
Figure 5-3: Comparison of R-learning, Q-learning and Q _M -learning with eligibility traces.	104
Figure 5-4: Approximated optimal value function with maximum reward FITSK-RL.	107
Figure 6-1: Graphical representation of error function.	129
Figure 6-2: Masking of SBP and MCES.	130
Figure 6-3: Average weights of the six features for the RPE problem.	131
Figure 6-4: MCES algorithm with ANFIS model on the MPG prediction problem.	132
Figure 6-5: Simulation of ANFIS with different input feature set.	133
Figure 6-6: Average weight and ranking of MCES on the audio detection problem.	134
Figure 6-7: Performance comparison of the feature selection algorithms.	135
Figure 6-8: Benchmarking results for the stock prediction problem with MCES.	137
Figure 6-9: Performance comparison with different combinations of features.	137
Figure 7-1: A blended view of research areas in this Thesis.	140

List of Tables

Table 2-1: Summary of convergence results. 19

Table 2-2: Survey of Fuzzy RL models..... 29

Table 3-1: Specifications of tasks 51

Table 3-2: Performance comparison for task (a). 53

Table 3-3: Performance comparison for task (b). 53

Table 3-4: Performance comparison for task (c). 55

Table 3-5: Relative MSE for three tasks..... 55

Table 3-6: Relative Pearson correlation coefficient for three tasks. 55

Table 3-7: Testing results of Mackey-Glass series. 57

Table 4-1: Predefined FITSK-RL parameters..... 81

Table 5-1: Predefined FITSK-RL parameters..... 107

Table 5-2: Results of the optimal stopping problem..... 108

Table 6-1: Dimensions of Feature Selection Algorithms for Regression Tasks..... 122

Table 6-2: Six features of the automobile MPG prediction problem..... 132

Table 6-3: Features selected with different feature selection algorithms. 135

Table 7-1: A summarized view of research achievements in this Thesis. 140

List of Abbreviations

AHC	Adaptive heuristic critic
ANFIS	Adaptive-network-based fuzzy inference systems
CMAC	Cerebeller model articulation controller
FITSK	Fuzzy input Takagi-Sugeno-Kang fuzzy framework
FITSK ⁰	Zero-order FITSK model
FITSK ¹ -GKF	First-order FITSK model with global Kalman filter learning
FITSK ¹ -LKF	First-order FITSK model with localized version of Kalman filter learning
FITSK-RL	Fuzzy input Takagi-Sugeno-Kang for reinforcement learning
FRL	Fuzzy reinforcement learning
GMP	Generalized modus ponens
GMT	Generalized modus tollens
KCN_MSE	K-central-nearest mean-squared-error
LWR	Local weighted regression
MCES	Monte Carlo evaluative selection
MDP	Markov decision process
MIMO	Multiple-input-multiple-output
MLP	Multi-layered-perceptron
MSE	Mean-squared-error
RL	Reinforcement learning
RMSE	Root- mean-squared-error
SBP	Sensitivity-based pruning
TD	Temporal difference
TSK	Takagi-Sugeno-Kang fuzzy model

Fuzzy Modelling in Reinforcement Learning

by
Quah Kian Hong

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy
Oct 2005

Summary

A *fuzzy reinforcement learning* model is an integration of fuzzy system to implement the role of *value function approximation* in reinforcement learning. Existing successful value function approximations can be broadly attributed to classes of local basis function approximation methods. However, the local basis function methods lack a systematic and definite procedure on the crafting of the basis function. On the contrary, classes of fuzzy reinforcement learning models are defined systematically by employing some fuzzy inference schemes. However, the roles of the fuzzy systems in these existing models are weakly described; and these models are not modularly formulated. On the other hand, existing *Takagi-Sugeno-Kang* (TSK) fuzzy models attempt to optimize the global learning accuracy as well as to maintain the interpretability of the local model in nonlinear system estimation. Most of the existing methods suffer from the use of offline learning algorithms to globally optimize this multi-criteria problem. Despite the ability to reach an optimal solution in terms of accuracy and interpretability, these offline methods are not applicable to online reinforcement learning system. Furthermore, most of the learning methods in TSK model are susceptible to the curse-of-dimensionality limitation. Therefore, it is crucial to design an online TSK framework with the following design considerations; namely: the interpretability of the local model; the global accuracy; and the system dimensionality issues.

A generic *Fuzzy Input Takagi-Sugeno-Kang* fuzzy framework (FITSK) is proposed to handle the different scenarios in this design problem. The online learning FITSK framework is extensible to both the zero-order and the first-order FITSK models. A localized version of Kalman filter algorithm is proposed for the parameter tuning of the first-order FITSK model.

It achieves a balance between interpretability and learning accuracy of a system, and possesses greater immunity to the curse-of-dimensionality. Subsequently, the zero-order FITSK model is integrated into a proposed *Generic Reinforcement Learning Framework*. This novel fuzzy reinforcement learning model is termed the *Fuzzy Input Takagi-Sugeno-Kang for Reinforcement Learning* (FITSK-RL) model. This ensures that the FITSK-IU model is systematically defined, able to be applied to continuous problem domains, and has strong theoretical foundations based on the fuzzy inference and the local basis function perspectives. This model is demonstrated to generalize three popular temporal learning algorithms based on the generic reinforcement learning framework. Subsequently, a *Maximum Reward Reinforcement Learning* algorithm is proposed to comprehend the learning of non-cumulative reward reinforcement learning problem. This novel temporal learning algorithm is then mapped into the generic reinforcement learning framework, and is demonstrated to effectively model an optimal stopping problem in financial derivative pricing. Finally, the *Monte Carlo Evaluative Selection* (MCES) approach is proposed as an orthogonal approach in complementing the curse-of-dimensionality issue in the design of TSK fuzzy system. These frameworks and models are evaluated using an extensive suite of simulations and applications in this Thesis. The experimental results are encouraging.

Thesis Supervisor: Dr Quek Hiok Chai

Title: Associate Professor in Division of Computer Science, School of Computer Engineering

1 Introduction

*"The Great River rolls to sea, flowers do waves thrash, heroes do sands smash.
When all the dreams drain, same are loss and gain: Green mountains remain, under pink sunsets."
~ Luo Guanzhong*

1.1 Motivations

Reinforcement learning is a model-free computational learning approach (Section 2.1.2) to map situations to actions, which attempts to maximize total future rewards or long-term goals [Kaelbling 1996], [Sutton 1998]. Reinforcement learning model often involves two learning problems, which are the *temporal credit assignment problem* and the *structural credit assignment problem* [Sutton 1984]. The temporal credit assignment problem tackles the issue of assigning delayed credit or blame to states or actions in a temporal sequence, and has been well studied in several research efforts [Sutton 1988], [Watkins 1989], [Rummery 1995], [Sutton 1998]. The structural credit assignment involves the process of propagating the awarded credit or blame in a particular state or action, to be stored in some internal structures or memories.

However, due to Bellman's *curse of dimensionality and modeling* [Bellman 1957] (Section 2.1.3), the usage of simple tabular storage as the internal structure is inadequate. Therefore, it is crucial to devise a function approximator to generalize the return value function during the bootstrapping learning process [Bertsekas 1996] (Section 2.1.3). However, early attempts to generalize the value function have encountered significant failures in practice [Boyan 1995a], [Baird 1995], [Gordon 1995a]. On the contrary, some research efforts demonstrated that value function approximations are still possible in practice [Tsitsiklis 1996], [Sutton 1996b], and in theory [Tsitsiklis 1997], [Van Roy 1998]. These successful examples employ *feature-based methods* or *local basis functions* as the function approximator [Tsitsiklis 1996], [Sutton 1996b], [Tsitsiklis 1997], [Van Roy 1998]. However, the local basis function methods lack a systematic and definite procedure on the crafting of the basis function [Van Roy 1998]. On the contrary, classes of *fuzzy reinforcement learning* models are defined systematically by employing some fuzzy inference schemes [Berenji 1992], [Glorennec 1994], [Berenji 1996], [Lin 1996a], [Jouffe 1998]. However, the roles of the fuzzy systems in these models are weakly described and are not modularly formulated. Therefore, the existing fuzzy reinforcement learning models are not applicable to general classes of reinforcement learning algorithms. Hence, it is desirable to devise a novel fuzzy reinforcement learning model that integrates the strengths of local basis function and fuzzy models.

Apart from these, existing Takagi-Sugeno-Kang (TSK) fuzzy models [Sugeno 1988], [Takagi 1985] proposed in the literature attempt to optimize the global learning accuracy as well as to maintain the interpretability of the local model [Yen 1998], [Abonyi 2000], [Johansen 2000], [Johansen 2003]. However, most of the proposed methods suffer from the use of offline learning algorithms to globally optimize this multi-criteria problem [Yen 1998], [Johansen 2003]. Despite the ability to reach an optimal solution in terms of accuracy and interpretability, these offline methods are not applicable to learning in adaptive or incremental systems, such as the online learning in reinforcement learning. Furthermore, most of the learning methods in TSK model are susceptible to the limitation of the curse-of-dimensionality. Therefore, the following criteria are crucial in the design of TSK model; they are namely:

- the interpretability of the local model;
- the global accuracy; and
- the dimensionality of the system.

The design of a generic TSK framework with online learning capability to handle the different scenarios based on these criteria is a crucial modelling problem.

Back to the temporal credit assignment problem, existing reinforcement learning paradigms proposed in the literature are guided by two performance criteria; namely: the expected cumulative-reward, and the average reward criteria [Puterman 1994], [Sutton 1984], [Sutton 1988], [Watkins 1989], [Rummery 1994], [Kaelbling 1996], [Bertsekas 1996], [Sutton 1998]. Both of these criteria assume an underlying cumulative or additivity of the rewards. However, such underlying cumulative of the rewards is not a definite necessity in some contexts. Two possible scenarios are presented in Section 5, and are summarized as follows. The first concerns with learning of an optimal policy that is further away in the existence of a sub-optimal policy that is nearer. The cumulative rewards paradigms suffer from slower convergence due to the influence of accumulating the lower rewards, and take time to fade away the effect of the sub-optimal policy. The second scenario concerns with approximating the supremum values of the payoffs of an optimal stopping problem. The payoffs are non-cumulative in nature, and thus the cumulative rewards paradigm is not applicable to resolve this. Hence, a non-cumulative reward reinforcement learning paradigm is needed in these application contexts.

Finally, it is noted that in the design of fuzzy models, the curse-of-dimensionality remains a crucial problem despite efforts to remedy it. Furthermore, most recent research efforts on

feature selection have focused mainly on classification task due to its popularity in the data-mining community [Jang 1996], [Utans 1991], [Laar 1997], [Kwak 2002], [Mao 2002], [Setiono 1997], [Setiono 2002], [Lee 2001], [Hsu 2002], [Petersen 1998]. However, feature selection research in nonlinear system estimations has been very limited. Hence, it is reasonable to devise a feature selection approach that is computationally efficient on nonlinear system estimations context. This approach complements the dimensionality reduction efforts in devising an effective online fuzzy model.

1.2 Objectives

The objectives of this Thesis are to tackle the modelling issues of fuzzy systems, and subsequently solving the value function approximation problem in reinforcement learning with the proposed fuzzy models. Furthermore, a non-cumulative reward reinforcement learning paradigm is proposed to supplement the existing cumulative reward temporal credit assignment problem. Finally, an objective feature selection approach is devised to complement the research efforts in dimensionality reduction of nonlinear system estimations.

1.3 Major Contributions

In the context of solving the modelling issues described in Section 1.1, four major contributions have been achieved in this Thesis; they are namely:

- A novel fuzzy framework, the *generic fuzzy input Takagi-Sugeno-Kang fuzzy framework* (FITSK), is proposed in Section 3 of this Thesis. The FITSK framework is extensible to both the zero-order and the first-order FITSK models. A zero-order FITSK model is suitable for the learning of adaptive system, and the bias-variance of the system can be easily controlled through the degree of localization. On the other hand, a first-order FITSK model is able to achieve higher learning accuracy for nonlinear system estimation. A localized version of the Kalman filter algorithm is proposed for the parameter tuning of the first-order FITSK model. The localized Kalman filter is able to achieve a balance between interpretability and learning accuracy of a system, and possesses a greater degree of immunity to the curse-of-dimensionality. The learning algorithms for the FITSK models are online, and are readily applicable to adaptive system with fast convergence speed. Finally, a proposed guideline is discussed to handle the model selection of different FITSK models to tackle the multi-criteria design problem of applying the TSK-model. Extensive benchmarks were conducted using the proposed FITSK models and

their learning algorithms; their performances are encouraging when benchmarked against other popular fuzzy systems; such as ANFIS [Jang 1992], TSK⁰-FCMAC [Ting 2002], Sugeno's, Mamdani's and Turksen's models [Nakanishi 1993].

- A new fuzzy reinforcement learning model, the *Fuzzy Input Takagi-Sugeno-Kang for Reinforcement Learning* (FITSK-RL), is proposed in Section 4 of this Thesis. The design of FITSK-RL adopts a two-stage approach. A generic reinforcement learning framework that has strong modularity and generality is designed in the first stage. This generic framework provides functional independence of the modules within the framework. Subsequently, this framework is demonstrated to be able to generalize three popular reinforcement learning algorithms; namely: TD(λ) [Sutton 1988], SARSA [Sutton 1998], and Q-learning algorithms [Watkins 1989]. The second stage formulates the zero-order FITSK model to assume the function approximator role of the framework. This ensures that the FITSK-RL model is systematically defined, able to be applied to continuous problem domains, and has strong theoretical foundations based on the fuzzy inference and the local basis function perspectives. Simulations were conducted using the proposed FITSK-RL model. The performance is encouraging when applying it to the discrete state-action, continuous state-discrete action, and continuous state-action reinforcement learning tasks.
- A maximum reward criterion is proposed in Section 5 of the Thesis, and the resulting reinforcement learning model with this learning criterion is termed the *maximum reward reinforcement learning*. The maximum reward reinforcement learning considers the learning of non-cumulative rewards problem, where the agent (define in Section 2.1) exhibits a maximum reward-oriented behavior towards the largest rewards in the state-space. Intermediate lower rewards that lead to sub-optimal policies are ignored in this learning paradigm. The maximum reward reinforcement learning is subsequently modeled with the FITSK-RL model. Finally, the model is applied to an optimal stopping problem with a nature of non-cumulative rewards, and its performance is encouraging when benchmarked against Van Roy and Tsitsiklis model [Van Roy 1998], [Tsitsiklis 1999].
- A novel feature selection approach, the *Monte Carlo Evaluative Selection* (MCES), is proposed in Section 6 of the Thesis. MCES is an objective sampling method that derives an unbiased estimation of the relevancy measure. The algorithm is objectively designed to avoid the major limitations due to biases present in existing feature selection algorithms. The MCES method has been demonstrated to perform well with four sets of experiments,

consisting of an artificial dataset and three practical datasets. The experimental results demonstrate that the MCES method has strong advantages; namely:

- low computational cost;
- ability to identify correlated and irrelevant features based on weight ranking;
- applicable to both nonlinear system estimation and classification tasks; and
- is independent of the underlying induction algorithms used to derive the performance measures.

This partially alleviates the curse-of-dimensionality issue in the design of fuzzy models.

1.4 Organization of the Thesis

The subsequent sections of this Thesis are organized as follows. Section 2 presents a brief review of reinforcement learning and fuzzy systems with low technical rigor, to serve as a general understanding of these relevant topics. Section 3 presents the FITSK fuzzy framework and the design criteria pertaining to this fuzzy modeling problem. Section 4 devises a generic reinforcement learning framework, and applies the FITSK model for value function approximation of this generic framework. The result of this arrangement is the proposed FITSK-RL model. Section 5 describes the maximum reward reinforcement learning paradigm for the context of non-cumulative reward reinforcement learning. The maximum reward is further merged into the FITSK-RL model and is applied to an optimal stopping problem. Section 6 proposes the MCES approach to partially remedy the inherent curse-of-dimensionality issue in machine learning problems. Section 7 concludes this Thesis with recommendations for future improvements.

2 Review of Reinforcement Learning and Fuzzy Systems

"If I have seen further than others, it is by standing upon the shoulders of giants."
~ Isaac Newton

This section presents a review of some basic concepts in reinforcement learning and fuzzy systems. Issues and deficiencies in the existing reinforcement learning and fuzzy systems are highlighted and are elaborated. Finally, the fuzzy reinforcement learning models, which synergize the existing efforts of fuzzy systems and reinforcement learning are discussed. The reviews in this section are generic on topics directly related to the research in the subsequent sections; and thus, serving as the summary of all these relevant topics. Please refer to the referred references for an in-depth study of each subtopic.

2.1 Reinforcement Learning

A reinforcement learning paradigm is an aggregation of the following five elements; namely: the *agent*, the *environment*, the *state* information, the *reinforcement signal*, and the *action* [Kaelbling 1996].

Figure 2-1: A typical reinforcement learning paradigm.

As shown in Figure 2-1, an *agent* interacts with its *environment*. On each step of interaction, the agent perceives some observable input and *reinforcement signal*, which give some indication of the current *state* of the environment. Based on the current input and reinforcement signal, the agent generates an *action* as output. The state transition of the environment is determined by the action acting on it. The dynamics of the reinforcement learning system are described by the following three elements; namely: the *policy*, the *reward function*, and the *value function* [Sutton 1998].

At some specific time step, a perceived state is mapped to an action based on a *policy*. A policy defines the behaviour of the learning agent. It may constitute by using a simple function, a lookup table, or as complicated as an extensive computational search process. Policies are generally stochastic.

A *reward function* maps each perceived state to a reinforcement signal, or numerical reward. The reward function indicates the immediate favourable or unfavourable quality of the current state. The *value function* of a state specifies the favourable or unfavourable quality in the long run. It is usually defined as the total amount of reward an agent can expect to accumulate over the future time step, starting from the current state. The objective of a reinforcement learning system is to search for the best policy with the maximum value function.

Some sources of reinforcement learning literature include a brief description in [Sutton 1999], an introductory tutorial in [Harmon 1996], a survey by [Kaelbling 1996], a set of introductory course notes by [Sutton 1996a], an introductory textbook by [Sutton 1998], and a textbook with a formal mathematical treatment of the problem by [Bertsekas 1996].

2.1.1 Basic Mathematical Definitions

This subsection summarizes the basic definitions and mathematical notations of reinforcement learning from the literature, especially from the textbook by [Sutton 1998].

Figure 2-2: A sequence of Markov decision process.

Consider a sequence of an infinite horizon *Markov decision process* (MDP) [Freedman 1983], [White 1993], [Puterman 1994] with discrete time step $t=0,1,2,3,\dots,\infty$ (Figure 2-2). At each discrete time step, the agent receives some observable environment's states, $s_t \in S$, where S is the set of possible states. The agent implements a mapping from the state to probabilities of selecting each possible action, $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state s_t . The mapping is called the agent's policy, π_t , and is depicted by equation (2-1).

$$\text{Policy mapping; } \pi_t(s, a) = \Pr(a_t = a \mid s_t = s) \quad (2-1)$$

indicates the probability of choosing action a when the perceived state is s at time t . Due to the action a_t , the agent receives a numerical reward r_{t+1} , and observes a new state s_{t+1} at time $t+1$.

The goal of the learning algorithm is to maximize the successive total reward, or the *return* value as described in equation (2-2).

$$\text{Return; } R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2-2)$$

The discount rate, γ , where $0 \leq \gamma \leq 1$, determines the present value of future rewards, a reward received k time steps in the future is worth only γ^k times what it would be worth if it were received immediately. If $\gamma=0$, then the agent's actions only influence the immediate reward, r_{t+1} , as in most supervised learning agent. As γ approaches 1, the agent will take future rewards more seriously, or become more farsighted.

Given any state, s , and action, a , the probability of each possible next state, s' , or the *state transition probability* from state s to state s' is described by equation (2-3).

$$\text{State transition probability; } P_{ss'}^a = \Pr(s_{t+1} = s' | s_t = s, a_t = a) \quad (2-3)$$

Similarly, the expected value of the next reward is given by equation (2-4).

$$\text{Expected reward; } R_{ss'}^a = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s') \quad (2-4)$$

where $E(.)$ is the expectation function.

Given a policy, π , which maps state s to an action a , the *value* of the state s under policy π , denoted by $V^\pi(s)$, is defined as the expected *return* at state s by following the same policy at all subsequent states. Equation (2-5) is the *Bellman equation* for V^π , or the recursive version of the *state-value function for policy π* [Dayan 2001, Chapter 9].

$$\begin{aligned} \text{State-value under policy } \pi; V^\pi(s) &= E_\pi(R_t | s_t = s) \\ &= E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right) \\ &= E_\pi\left(r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right) \\ &= \sum_{\forall a \in A(s)} \pi(s, a) \sum_{\forall s' \in S'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')] \end{aligned} \quad (2-5)$$

where $E_\pi(.)$ is the expectation function with a given policy π .

Similarly, the *action-value function for policy π* is defined as the value of taking action a in state s under a policy π , denoted as $Q^\pi(s, a)$, is described by equation (2-6).

$$\begin{aligned}
 \text{Action-value under policy } \pi; Q^\pi(s, a) &= E_\pi(R_t | s_t = s, a_t = a) \\
 &= E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right) \\
 &= E_\pi(r_{t+1} + \gamma V^\pi(s') | s_t = s, a_t = a) \\
 &= \sum_{s' \in S'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')] \\
 &= \sum_{s' \in S'} P_{ss'}^a [R_{ss'}^a + \gamma \sum_{a' \in A(s')} \pi(s', a') Q^\pi(s', a')]
 \end{aligned} \tag{2-6}$$

Consider an *optimal policy*, π^* , where the value of such policy is the largest among all the state, as described in equation (2-7).

$$V^{\pi^*}(s) \geq V^\pi(s), \text{ for all } s \text{ and } \pi. \tag{2-7}$$

Then, an optimal policy has the highest state-value, and the *optimal state-value function*, V^* , is defined as in equation (2-8).

$$\begin{aligned}
 \text{Optimal state-value; } V^*(s) &= \max_{\pi} V^\pi(s) \\
 &= \max_a Q^{\pi^*}(s, a) \\
 &= \max_a \sum_{s' \in S'} P_{ss'}^a [R_{ss'}^a + \gamma V^*(s')]
 \end{aligned} \tag{2-8}$$

Similarly, the *optimal action-value function*, Q^* , is defined in equation (2-9).

$$\begin{aligned}
 \text{Optimal action-value; } Q^*(s, a) &= \max_{\pi} Q^\pi(s, a) \\
 &= E(r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a) \\
 &= E(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a) \\
 &= \sum_{s' \in S'} P_{ss'}^a [R_{ss'}^a + \gamma \max_{a'} Q^*(s', a')]
 \end{aligned} \tag{2-9}$$

Equations (2-8) and (2-9) are *Bellman optimality equations* [Bellman 1957] for V^* and Q^* .

Reinforcement learning methods are best explained based on the theoretical foundation of Markov decision processes [White 1993], [Puterman 1994]. Singh's thesis [Singh 1994] provides a good linkage between reinforcement learning, stochastic approximation theory, and finite stationary Markov decision processes. Littman [Littman 1996] went a step further

in generalizing some reinforcement learning methods based on a generalized Markov decision process model. Doya [Doya 2000] extends the reinforcement learning to the continuous domain with the mathematical treatments from the continuous time and state Markov decision processes.

2.1.2 Reinforcement Learning Models

There are two distinct classes of approaches to solve the Markov decision process (MDP) problems discussed in Section 2.1.1; namely: the model-based, and the model-free approaches (Figure 2-3). In equations (2-8) and (2-9), the optimality equations are computed by approximating the value functions and modeling the state transition probabilities. The state transition probabilities represent the model of the environment, which is stochastic. The model-based MDP approaches learn the state transition probabilities of each state, and simultaneously approximate the value functions with dynamic programming algorithms based on the state transition probabilities. In contrast, the model-free MDP approaches approximate the value functions directly, without the need of modeling the state transition probabilities prior to the approximation of value functions. Reinforcement learning models are the result of these model-free MDP approaches. These model-free approaches are made possible with the invention of the *temporal difference* (TD) learning [Sutton 1988]. The TD learning is a generalization of dynamic programming and Monte-Carlo methods [Sutton 1998]. Monte-Carlo method is an online sampling method without the need of having a model of the environment. The TD learning is relaxed from the need of computing the state transition probabilities by adopting update formulae similar to those of the Monte-Carlo method.

Figure 2-3: Approaches to solve Markov decision processes.

These approaches are further classified into the applications in the autonomous and controlled systems. The autonomous systems are applications whereby the system’s dynamics are not

influenced by the control decisions. In contrast, the controlled systems require the intervention of the control decisions to influence the system's dynamics [Van Roy 1998]. The temporal difference (TD) learning method [Sutton 1988] is a popular reinforcement learning method to model the model-free autonomous systems. The two most popular methods in modeling of model-free controlled systems are the *Q-learning* method [Watkins 1989], and the *SARSA learning* [Rummery 1994], [Sutton 1998]. Q-learning is an off-policy method, whereby the policy to generate the actual action and the policy to be estimated are being segregated. In contrast, SARSA learning (or modified Q-learning) is an on-policy method, whereby the actions are generated and are estimated based on the same policy. The following context in this subsection will mathematically define the TD learning, Q-learning and SARSA learning models.

(a) Temporal difference learning and actor-critic model

The temporal difference (TD) learning is adopted for autonomous system whereby the control decisions are independent of the underlying system dynamics. Thus, the objective of the learning is to approximate the optimal state-value function (equation (2-8)). The update formula for the estimation of a state-value can be represented as equation (2-10).

$$V_{t+1}(s_t) = V_t(s_t) + \alpha[R_t - V_t(s_t)] \quad (2-10)$$

The actual return, R_t , needs to be presented before proceeding with the actual update. However, the actual return is only available at the end of the learning episode. Therefore, the TD learning is introduced to estimate the actual return based on the immediate reward and the discounted next state-value. Thus, the update rule of equation (2-10) can be modified into equation (2-11).

$$V_{t+1}(s_t) = V_t(s_t) + \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)] \quad (2-11)$$

This method is the simplest form of TD learning, the one-step temporal-difference (TD(0)) method. The method does not require the process to experience the end of the learning episode before proceeding with the update. The update is only dependent on the immediate reward and the next state value.

The TD(λ) method is the generic form of TD learning, where $\lambda \in [0,1]$ refers to the *recency factor*, or the use of *eligibility traces* [Sutton 1984] in actual learning. It measures how recent a state is being visited. This is represented with eligibility traces, and the decaying effect of

the traces is controlled by the recency factor. TD(λ) methods are generalization of TD(0) to Monte Carlo sampling methods (TD(1)). Sutton [Sutton 1988] shown that the TD(λ) learning performs equally-well to Widrow-Hoff supervised learning [Widrow 1960] whenever λ equals to one. However, the performance is better than the conventional Widrow-Hoff supervised learning algorithm for any general λ .

Extending equation (2-2) such that the return, R_t , is being represented with the future rewards and a future approximated state-value, as shown in equation (2-12).

$$\begin{aligned} R_t^{(1)} &= r_{t+1} + \mathcal{V}_t(s_{t+1}), \\ R_t^{(2)} &= r_{t+1} + \mathcal{V}_{t+2} + \gamma^2 V_t(s_{t+2}), \\ &\vdots \\ R_t^{(n)} &= r_{t+1} + \mathcal{V}_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}) \end{aligned} \quad (2-12)$$

$R_t^{(n)}$ is the n^{th} -step return at time t , it is a return truncated after n steps and then approximately corrected for the truncation by adding the estimated value of the n th next state. Each $R_t^{(n)}$ is an n^{th} -step approximation of R_t .

A weighted average of R_t , with each of the n -step return weighted proportionally to $(1-\lambda)\lambda^{n-1}$, is shown in equation (2-13).

$$R_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} \quad (2-13)$$

For a sequence of states where the terminal state occurred at the T^{th} time-step, all the subsequent n -step returns following that terminal state are equal to R_t . Thus, equation (2-13) can be decomposed into equation (2-14).

$$\begin{aligned} R_t^\lambda &= (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} \\ &= (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + (1-\lambda) \sum_{n=T-t}^{\infty} \lambda^{n-1} R_t \\ &= (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t (1-\lambda) \sum_{n=0}^{\infty} \lambda^n \\ &= \underbrace{(1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)}}_{\text{Return functions till the terminal state}} + \underbrace{\lambda^{T-t-1} R_t}_{\text{Subsequent return function after the terminal state}} \end{aligned} \quad (2-14)$$

Equation (2-15) is formulated by considering a slightly different version of equation (2-10), the estimation error of the state-value function of state, s_t at time t , is adjusted based on this weighted average return value.

$$\Delta V_t(s_t) = \alpha[R_t^\lambda - V_t(s_t)] \quad (2-15)$$

Equations (2-14) and (2-15) shows that, when $\lambda=0$, the update takes the form of the TD(0) learning; when $\lambda=1$, the update is identical to the Monte Carlo sampling learning method.

In actual implementation, the update of equation (2-15) involves the usage of *eligibility trace*. The eligibility traces formulae were proposed by Sutton [Sutton 1984], on his thesis on solving temporal credit assignment problem. The eligibility traces require higher computational resources than the one-step methods. However, they significantly speed up the learning process; particularly when rewards are delayed for long time ahead in the future.

An *accumulating eligibility trace* formula for state s at time t is shown in equation (2-16).

$$e_t(s) = \begin{cases} \gamma \lambda e_{t-1}(s) & , \quad s \neq s_t \\ \gamma \lambda e_{t-1}(s) + 1 & , \quad s = s_t \end{cases} \quad (2-16)$$

This eligibility trace is an accumulation of traces leaving behind in previous visited states. The trace will gradually fade away when the state is not being revisited. The decaying effect is dependant on the recency factor and the discount rate parameter, as defined by $\gamma \lambda$. It indicates the degree of eligibility for each state to undergo learning changes for future event. A possible problem of this accumulating eligibility trace arises when a state is revisited before the trace has decayed to zero. The revisited state will have a greater than one trace in this scenario. Singh and Sutton [Singh 1996] show that the accumulating eligibility trace formula resulted in a biased estimation of the value function. Thus, they proposed the *replacing eligibility trace* formula, which resulted in an unbiased learning of the value function (equation (2-17)).

$$e_t(s) = \begin{cases} \gamma \lambda e_{t-1}(s) & , \quad s \neq s_t \\ 1 & , \quad s = s_t \end{cases} \quad (2-17)$$

The TD error for state s_t at time t is represented in equation (2-18).

$$\delta_t(s_t) = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \quad (2-18)$$

where $\delta_t(s_t)$ is the TD error for state s_t at time t . Then the estimation error for the state-value function is shown in equation (2-19).

$$\Delta V_t(s) = \alpha \delta_t(s) e_t(s), \forall s \in S \quad (2-19)$$

Sutton [Sutton 1998] shows that equations (2-15) and (2-19) are equivalent. The computational cost of the eligibility trace formulae is reducible because most of the time the eligibility trace is zero, other than those recently visited states. The eligibility trace formulae offer an online incremental mean of learning method for the general TD learning.

Actor-critic model [Sutton 1998] or *adaptive heuristic critic* (AHC) algorithm [Sutton 1984] is the first method to extend the TD learning in reinforcement learning to controlled system. The actor-critic model segregates the value function approximation and the policy generation process. The policy selector is known as the *actor* or *action selector*; the state-value function estimator is known as the *critic*, to critique the actions selected by the actor. The critic learns and critiques on the current policy selected by the actor. The actor-critic model is depicted in Figure 2-4.

Figure 2-4: Actor-critic model

The function of the critic is to learn a state-value function using the TD learning formula (equation (2-18)). For each taken action, the critic evaluates the new state with the TD error. The TD error is subsequently feedback to the critic for refinement using the TD learning formula. If an error is positive, then the action being selected at time t , a_t , should have a higher probability of being selected in the future, and vice-versa. Supposingly, the actions are generated by the *Gibbs softmax method*, as shown in equation (2-20).

$$\pi_t(s, a) = \frac{e^{p_t(s, a)}}{\sum_{\forall b \in A(s)} e^{p_t(s, b)}} \quad (2-20)$$

where $p(s, a)$ are the modifiable policy parameters of the actor, indicating the tendency to select each action a while in each state s . It can be computed with equation (2-21).

$$p_{t+1}(s, a) = p_t(s, a) + \alpha \delta_t e_t(s, a) \quad (2-21)$$

where,

$$e_t(s, a) = \begin{cases} \gamma \lambda e_{t-1}(s, a) + 1 - \pi_t(s_t, a_t), & s = s_t \text{ \& } a = a_t \\ \gamma \lambda e_{t-1}(s, a), & \text{otherwise} \end{cases} \quad (2-22)$$

Some advances in actor-critic model and its convergence proof can be found in [Konda 2003].

(b) Q-learning model

A controlled system is an active system whereby the system's dynamics are directly influenced by the control decisions and policies [Van Roy 1998]. There are two classes of controlled reinforcement learning models; namely: the on-policy and the off-policy TD models. The on-policy model uses an identical policy to generate control behavior and to estimate the optimal policy. On the contrary, the off-policy model assumes different *behavior* and *estimation* policies [Sutton 1998]. Q-learning [Watkins 1989], [Watkins 1992] is a popular off-policy model-free reinforcement learning model. The estimation policy of Q-learning is a greedy policy that constantly searches for the best action-value. This estimation policy may be different from the behavior policy in off-policy model. In contrast, the SARSA learning [Rummery 1994] is a popular on-policy model-free model, where the estimation and behavior policies are identical. Both models approximate the optimal action-value (equation (2-9)) directly without computation of the state-value.

Q-learning is an off-policy TD control model following the general structure of policy iteration. The one-step Q-learning is defined as in equation (2-23).

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_{\forall a' \in A(s_{t+1})} Q_t(s_{t+1}, a') - Q_t(s_t, a_t)] \quad (2-23)$$

The action-value function, Q , directly approximates Q^* , the optimal action-value function, independent of the policy being followed.

The eligibility trace formulae for $Q(\lambda)$ learning, which is the Q-learning with a generally defined recency factor, λ , is updated with equation (2-24).

$$\begin{aligned} \text{Accumulating: } e_t(s, a) &= \begin{cases} \gamma \lambda e_{t-1}(s, a) + 1 & , \quad \forall s = s_t, a \text{ is greedy} \\ \gamma \lambda e_{t-1}(s, a) & , \quad \forall s \neq s_t, a \text{ is greedy} \\ 0 & , \quad a \text{ is non-greedy} \end{cases} \\ \text{Replacing: } e_t(s, a) &= \begin{cases} 1 & , \quad \forall s = s_t, a \text{ is greedy} \\ \gamma \lambda e_{t-1}(s, a) & , \quad \forall s \neq s_t, a \text{ is greedy} \\ 0 & , \quad a \text{ is non-greedy} \end{cases} \end{aligned} \quad (2-24)$$

The update formulae for $Q(\lambda)$ learning is represented with equation (2-25).

$$\begin{aligned} \delta_t &= r_{t+1} + \gamma \max_{a' \in A(s_{t+1})} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \\ Q_{t+1}(s, a) &= Q_t(s, a) + \alpha \delta_t e_t(s, a), \forall s \in S, \forall a \in A(s) \end{aligned} \quad (2-25)$$

The convergence of Watkins' Q-learning has been formally proven in his technical note [Watkins 1992] and Jaakkola's paper [Jaakkola 1994], provided that all actions are repeatedly sampled in all states and the actions-values are represented discretely. Peng and William [Peng 1996] went a step further by introducing a modified version of $Q(\lambda)$. Empirically, Peng's method performs better than $Q(\lambda)$, although the convergence has not yet been theoretically proven. Dayan [Dayan 1992] used Q-learning techniques to prove the convergence of temporal difference methods in the mean for general λ and weaken their reliance on Markovian environment assumption. Therefore, the convergence of tabular-based Q-learning model (without using function approximation of action-value) is guaranteed.

(c) SARSA model

SARSA or *modified Q-learning* [Rummery 1994], [Sutton 1996b] is a variation of on-policy model-free control approach following the pattern of policy iteration. This on-policy model evaluates and improves the policy, which is eventually used for control decisions.

The eligibility trace formulae of SARSA learning are given by equation (2-26).

$$\begin{aligned} \text{Accumulating: } e_t(s, a) &= \begin{cases} \gamma \lambda e_{t-1}(s, a) + 1, & s = s_t \text{ \& } a = a_t \\ \gamma \lambda e_{t-1}(s, a), & \text{otherwise} \end{cases} \\ \text{Replacing: } e_t(s, a) &= \begin{cases} 1, & s = s_t \text{ \& } a = a_t \\ \gamma \lambda e_{t-1}(s, a), & \text{otherwise} \end{cases} \end{aligned} \quad \text{, for all } s \in S, a \in A(s) \quad (2-26)$$

The update rule for the algorithm is given in equation (2-27).

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \delta_t e_t(s, a), \text{ for all } s \in S, a \in A(s) \quad (2-27)$$

where the TD error is computed with equation (2-28).

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \quad (2-28)$$

The SARSA learning has slower learning performance than the Q-learning model due to the usage of non-greedy estimation policy. However, Sutton [Sutton 1996b] has shown that the SARSA learning has a better position than Q-learning in terms of convergence. This happens when function approximations are being applied to both models. Van Roy [Van Roy 1998] further proved that off-policy models, such as Q-learning, are generally diverging in the presence of function approximation for action-value estimations.

2.1.3 Function Approximations in Reinforcement Learning

Reinforcement learning model often involves two learning problems, which are the temporal credit assignment problem and the structural credit assignment problem [Sutton 1984]. The temporal credit assignment problem tackles the issue of assigning delayed credit or blame to states or actions in a temporal sequence; this has been reviewed in Section 2.1.2. The structural credit assignment involves the process of propagating the awarded credit or blame in a particular state or action, to be stored in some internal structures or memories. However, due to Bellman's *curses of dimensionality and modeling* [Bellman 1957], the usage of simple tabular storage as the internal structure is inadequate. The curses describe that the modelling complexities increase exponentially with respect to the linear increment in system dimensionality. Thus, it is difficult to model high dimensionality system appropriately with simple tabular storage. On the other hand, the reinforcement learning algorithm often assumes a bootstrap learning process, which means that the next time-step of approximated values is computed based on the prior approximated values. Therefore, it is crucial to devise a function approximator to generalize the return value function during this bootstrapping learning process [Bertsekas 1996].

The function approximator is a system to predict the action-value or state-value vector with a given state vector and a tunable parameter vector. The general idea of the function

approximator is to implement the function mapping of, $f_{FA} : S \times \mathbb{R}^K \mapsto \mathbb{R}^{|A(s)|}$, as described by equation (2-29).

$$Q_t(s_t, a) = f_{FA}(s_t, c_t), \forall a \in A(s_t) \quad (2-29)$$

where

- t is the time index,
- K is the number of tunable parameters of the function approximator,
- $f_{FA}(\cdot)$ is the function approximator's output,
- $c_t \in \mathbb{R}^K$ is the tunable parameter vector of the function approximator, and
- $Q_t(\cdot)$ is the action-value vector with cardinality $|A(s_t)|$ at time t .

Consequently, the tunable parameters are updated with respect to the state information and the temporal difference error, and is described by equation (2-30).

$$\Delta c_t = f_c(s_t, \delta_t) \quad (2-30)$$

where $f_c(\cdot)$ implements the error function of the tunable parameter.

Sutton [Sutton 1989] firstly proposed a general temporal learning algorithm to be approximated with connectionist approaches in an online manner. Other works include the REINFORCE algorithm [Williams 1992] and Lin's approaches [Lin 1992]. However, the REINFORCE algorithm deals mainly with immediate reward situations, and both approaches are implemented without eligibility traces. On the other hand, such generalization methods with gradient-descent learning have been known to diverge to infinite *mean-squared-errors* (MSE) for most of the control methods [Boyan 1995a], [Baird 1995], [Gordan 1995a], [Tsitsiklis 1996]. This is especially true in the case when bootstrapping (updating of a value estimate on the basis of other value estimates) is involved for TD learning. *Residual-gradient methods* [Baird 1995] and *Value and Policy Search* (VAPS) [Baird 1999a], [Baird 1999b] have been proposed to remedy this issue for *multi-layered-perceptron* (MLP) approximators. These methods have guaranteed convergence even when the exploration policy changes during learning. However, the convergence to global optimum remains to be a problem for these nonlinear function approximators, such as back-propagation neural networks [Rumelhart 1986a], [Rumelhart 1986b]. On the other hand, a control problem with on-policy method (SARSA) and *cerebeller model articulation controller* (CMAC) [Albus 1975] as the function approximator has been empirically shown to have good convergence results [Sutton 1996b].

Boyan and Moore [Boyan 1995a] performed a comparative study of convergence results with four different function approximators, i.e., linear regression, quadratic regression, local weighted regression (LWR) [Cleveland 1988], [Moore 1997], and MLPs. Simulations are performed on three problems; namely: a 2-D gridworld, a 2-D puddleworld, and a mountain-car problem. Four possible behaviors are studied for these function approximators; namely:

Good convergence: The function approximator accurately represents the intermediate value functions during the iterative learning process, and successfully converges to the eventual optimal value function.

Lucky convergence: The function approximator does not accurately represent the intermediate value functions during the iterative learning process, but manages to converge to a value function whose greedy policy is optimal.

Bad convergence: The algorithm converges, but the resulting value function and policy are poor.

Divergence: The temporal learning error is magnified from an iteration to the next; the value function diverges out of bound.

The result of their studies is summarized in Table 2-1 by classifying these into the four possible behaviours that have been described.

Table 2-1: Summary of convergence results.

Domain	Linear	Quadratic	LWR	MLPs
2-D gridworld	Lucky	Diverge	Good	Lucky
2-D puddle world	—	—	Diverge	Diverge
Mountain-Car	—	—	Good	Diverge

Boyan and Moore [Boyan 1995a] empirically demonstrated that the LWR method has superior immunity against the divergence issue to the MLPs in approximation of action-value functions with Q-learning. However, they did not theoretically point out the reason of divergence, nor suggested any possible reason leading to the divergence. They proposed to use the Grow-Support algorithm that “rollouts” from costs of actual paths to the goal, and not from the values of the previous iteration’s function approximated values. This method prevented the divergence, but it needed a model of the task dynamics such that it could apply dynamic programming backups in an offline manner to a fixed set of states.

Sutton [Sutton 1996b] effectively solved this problem by using the CMACs [Albus 1975] with multiple overlapping tilings of the state space. This method has been shown to converge for all the three problems studied by Boyan and Moore [Boyan 1995a], and a complicated

acrobat task. Contrary to Boyan and Moore model with an off-policy Q-learning method; an on-policy, model-free SARSA learning model was adopted by Sutton in his simulations. All the simulations have shown promising convergence results. However, no theoretical reason for such convergence was presented.

The theoretical aspect of the convergence issue was presented in Baird's paper [Baird 1995] and his thesis [Baird 1999b]. A counter-example was presented in his thesis as shown in Figure 2-5. Consider a Markov chain value iteration problem with seven states; the value of each state is given by the linear combination of two weights. Every transition yields a reinforcement of zero. Each possible transition is visited with equal probability. The function approximation is a simple lookup table, with one additional weight, w_0 , responsible for the generalization capability. For this problem, if weight w_0 were not used, then the problem reduces to a simple lookup table, with all the weights and values converging to zero. However, with the inclusion of weight w_0 , and if all weights are initially positive, then all of the values will unboundedly increase with a simple value iteration algorithm.

Figure 2-5: The 7-star problem for value iteration.

Several learning methods have been proposed to remedy this divergence issue, including the advantage updating [Baird 1993a], [Baird 1993b], [Baird 1994], and the residual algorithm [Baird 1995], [Baird 1999b]. However, these algorithms have exhibited a slower convergence rate. Furthermore, such algorithms are only applicable to MLP approximators. Similar divergence has been shown with CMAC approximators with off-policy learning [Gordon 1995a].

Function approximation or value generalization is crucial to solving the curse-of-dimensionality and modelling issues inherently present in reinforcement learning. Furthermore, function approximation provides a means of modelling the reinforcement learning problem in continuous state-space domain. However, the problem is so crucial and

complex that a workshop in the 12th International Conference on Machine Learning (ICML95) was dedicated to solve this value function approximation problem [Atkeson 1995], [Baird 1995], [Boyan 1995b], [Crites 1995], [Dreyfus 1995], [Gordon 1995b], [Littman 1995], [Moore 1995], [Sutton 1995], [Zhang 1995].

Finally, Tsitsiklis and Van Roy [Tsitsiklis 1997], [Van Roy 1998] have proven the convergence and stability issues for reinforcement learning systems based on a linear local basis function approximation method, provided that certain assumptions are met (discuss in Section 4.2.2). However, the local basis function method lacks a systematic and a broadly defined guideline for the selection of basis functions. These issues will be further discussed in Section 4 of this Thesis, where a systematically defined fuzzy system is proposed for the approximation of action-value or state-value functions. This is possible by extending a fuzzy system as a special case of the local basis function approximator method. This subsection serves to provide the historical aspects on the studies of convergence and stability issues to value function approximation in reinforcement learning. Please refer to the various papers cited in the section for the in-depth discussions of the function approximation issues.

2.2 Fuzzy Systems

This subsection describes and defines the basic concepts and notations of fuzzy set theory and fuzzy logic. Subsequently, the general concept of fuzzy models is reviewed, and is followed by the discussion of two popular fuzzy models. The detailed mathematical treatments can be found in following books [Yen 1999], [Kaufmann 1985], [Dubois 2000], [Zimmerman 1985]. Finally, existing fuzzy reinforcement learning models in the literature are surveyed and are studied.

2.2.1 Fuzzy Sets, Fuzzy Operations, and Fuzzy Rules

Fuzzy sets can be defined as follow [Zadeh 1965], [Lee 1990a], [Lee 1990b]:

Let U be a collection of objects denoted generically by $\{u\}$, which could be discrete or continuous. U is called the universe of discourse and u represents a generic element of U . A fuzzy set F in a universe of discourse U is characterized by a membership function μ_F , which takes values in the closed interval $[0,1]$ namely, $\mu_F: U \rightarrow [0,1]$. A fuzzy set may be viewed as a generalization of the concept of an ordinary set whose membership function only takes on two values $\{0,1\}$. Thus, a fuzzy set F in U may be represented as a set of ordered pairs of a generic element u and its grade of membership function: $F = \{(u, \mu_F(u)) | u \in U\}$.

The *support* of a fuzzy set F is the crisp set of all points u in U such that $\mu_F(u) > 0$. The element u in U at $\mu_F = 0.5$ is called the *crossover* point, and at $\mu_F = 1.0$ is called the *kernel* or *core* points. A fuzzy set whose support is a single point in U with $\mu_F = 1.0$ is referred to as a *fuzzy singleton*.

Corresponding to the some of the classical logic operations, like AND, OR and COMPLEMENT operations; fuzzy sets have a corresponding set of fuzzy logic operations, which are the *conjunction*, *disjunction* and *complement* operations. Let A and B be two fuzzy sets in U with membership functions μ_A and μ_B , respectively. The fuzzy operations are shown on the following equations:

$$\begin{aligned} \text{Conjunction of } A, B: \mu_{A \cap B}(u) &= \min\{\mu_A(u), \mu_B(u)\}; \\ \text{Disjunction of } A, B: \mu_{A \cup B}(u) &= \max\{\mu_A(u), \mu_B(u)\}; \\ \text{Complement of } A: \mu_{\bar{A}}(u) &= 1 - \mu_A(u). \end{aligned} \quad (2-31)$$

Among these fuzzy logic techniques, the fuzzy if-then inference rule (or fuzzy rule) is the most significant because of its success in many applications. Basically, there are two important fuzzy implication inference rules named the generalized modus ponens (GMP) and the generalized modus tollens (GMT), as shown below:

(GMP)	Premise 1: x is A' , <u>Premise 2: if x is A then y is B,</u> Consequence: y is B'
(GMT)	Premise 1: y is B' , <u>Premise 2: if x is A then y is B,</u> Consequence: x is A'

A fuzzy rule is the basic unit for capturing knowledge in many systems. Generally, it has the form of:

IF <antecedent> THEN <consequent>

The antecedent describes a condition, and the consequent describes a conclusion that can be drawn when the condition holds. The consequent of fuzzy rules can be classified into three categories:

Crisp consequent: IF...THEN $y=a$, where a is a crisp numeric value.

Fuzzy consequent: IF...THEN y is A , where A is a fuzzy set or a symbolic value.

Functional consequent: IF (x_1 is A_1 AND x_2 is A_2 AND... x_n is A_n) THEN $y=f(x_i)$; where $f(.)$ is a function mapping.

Generally, fuzzy rules with a crisp consequent can be processed more efficiently. A rule with a fuzzy consequent is easier to understand and suitable for human intuitive reasoning. Rules with functional consequent can be used to approximate complex nonlinear behaviours with smaller set of rules.

2.2.2 Fuzzy Modelling

A model that describes a functional mapping relationship using a set of fuzzy mapping rules is a *fuzzy rule-based model*, or *fuzzy model* [Yen 1999]. A fuzzy model is a model that is obtained by fusing multiple *local models* that are associated with fuzzy subspaces of the given system states. A fuzzy subspace is a region whose boundary allows a gradual transition from a region to its neighboring regions. Normally, fuzzy models involve the following four processes; namely: fuzzy partitioning, mapping of fuzzy subspaces to local models, fusion of multiple local models, and defuzzification.

- **Fuzzy partition:** A classical partition of a space is a collection of disjoint subspaces whose union is the entire space. A fuzzy partition generalizes the classical partition so that the transition from one subspace into neighboring subspaces is gradual.
- **Mapping of fuzzy subspaces to local models:** Fuzzy subspaces are mapped to functions local to their partitions. Types of function depend on the consequent of the fuzzy model as described in the previous subsection, namely, crisp, fuzzy or functional consequents.
- **Fusion of local models:** Multiple local models are fused with interpolative reasoning into a global model. Each local model is weighted based on its matching degree to form an overall global decision. Such a formulation produces a smooth transition of the target concept to be approximated.
- **Defuzzification:** If the consequent part of the fuzzy model is fuzzy, then a defuzzification process is needed to interpret the quantitative summary of the linguistic terms. For crisp and functional consequent, the consequent calculation is simply based on the local weighting of the various local functions.

There are important two classes of fuzzy rule-based models; namely: the Mamdani model [Mamdani 1975], and the Takagi-Sugeno-Kang (TSK) model [Takagi 1985], [Sugeno 1988].

The Mamdani model uses a fuzzy consequent as the local function, while TSK model adopts the crisp or functional consequent. The functional consequent is normally a zero-order or a first-order polynomial function. Detailed mathematical definitions of these models could be referred to [Yen 1999] and [Lin 1996b]. Brief descriptions of these models are summarized in the following subsection. The Mamdani model is suitable for linguistic interpretation or human intuitive reasoning processes. In contrast, the TSK model is suitable for nonlinear function approximation regressive tasks.

(a) The Mamdani Model

Consider a mapping from an r -dimensional input spaces of $U_1 \times U_2 \times \dots \times U_r$ to W , then a Mamdani model's rule mapping takes the following form.

$$R_i: \text{IF } (x_1 \text{ is } A_{i1}) \text{ and } \dots \text{and } (x_r \text{ is } A_{ir}) \text{ THEN } y \text{ is } C_i$$

where

$x_j \quad j=1,2,\dots,r$ are the input variables, $x_j \in U_j$,

y is the output variable, $y \in W$,

A_{ij} are fuzzy sets for x_j , in U_j dimension space,

C_i are fuzzy sets for y , in W dimension space, and

R_i is the i -th fuzzy rules.

Given inputs of the form:

$$(x_1 \text{ is } A'_1), (x_2 \text{ is } A'_2), \dots, (x_r \text{ is } A'_r)$$

where $A'_1, A'_2, \dots, A'_j, \dots, A'_r$ are fuzzy subsets of $U_1, U_2, \dots, U_j, \dots, U_r$. A'_j can be an input fuzzy set or an input singleton in U_j which is presented to the Mamdani model. A_{ij} is generally predefined during the network structure identification stage based on some input space clustering or partitioning algorithm. The contribution of rule R_i to a Mamdani model's output is a fuzzy set whose membership function is computed by equation (2-32).

$$\mu_{C_i}(y) = (\alpha_{i1} \wedge \alpha_{i2} \wedge \dots \wedge \alpha_{ir}) \wedge \mu_{C_i}(y) \quad (2-32)$$

where,

$\wedge$ denotes the "min" operator,

α_i is the matching degree of rule R_i , and

$$\alpha_{ij} = \sup_{x_j} (\mu_{A'_j}(x_j) \wedge \mu_{A_{ij}}(x_j)) \quad (2-33)$$

is the matching degree between x_j and R_i 's condition about x_j .

The final output of the model is the aggregation of outputs from all rules using the “max” operator:

$$\mu_C(y) = \max\{\mu_{C_1}(y), \mu_{C_2}(y), \dots, \mu_{C_L}(y)\} \quad (2-34)$$

where

L is the total number of output membership functions.

Similar to the input clusters, L is determined by some output clustering or partitioning techniques. The output C is a fuzzy set, which can be defuzzified into a crisp output using some defuzzification techniques.

Some of the existing Mamdani-type of fuzzy systems are FALCON [Lin 1996b], POPFNN [Zhou 1996], and GenSoFNN [Tung 2002]. The research on these fuzzy systems is not on modifying the Mamdani’s model itself, but focuses on the input-output space clustering techniques and the system identification of the model.

(b) The TSK Model

The motivation of the TSK model is to reduce the number of rules required by the Mamdani model, and to increase the accuracy of the system; especially for complex and high-dimensional problems. To achieve this goal, the TSK model replaces the fuzzy sets in the consequent of the Mamdani rule model with a first order polynomial equation of the input variables. A general r -input one-output TSK model consists of rules in the following form.

$$\begin{aligned} R_i: & \text{ IF } (x_1 \text{ is } A_{i1}) \text{ and } \dots \text{ and } (x_r \text{ is } A_{ir}) \\ & \text{ THEN } y = f_i(x_1, x_2, \dots, x_r) = b_{i0} + b_{i1}x_1 + \dots + b_{ir}x_r \end{aligned}$$

where,

f_i is a first order polynomial function, and

$b_{ij} \quad j=0,1,\dots,r$ are real-valued parameters.

The inference performed by the TSK model is an interpolation of all the relevant local models based on the degree of relevance of the input data to the fuzzy subspaces associated with the local models. The total output of the model is given by equation (2-35).

$$y = \frac{\sum_{i=1}^L \alpha_i f_i(x_1, x_2, \dots, x_r)}{\sum_{i=1}^L \alpha_i} = \frac{\sum_{i=1}^L \alpha_i (b_{i0} + b_{i1}x_1 + \dots + b_{ir}x_r)}{\sum_{i=1}^L \alpha_i} \quad (2-35)$$

where α_i is the matching degree of rule R_i , which is analogous to the computation of the matching degree of the Mamdani model, as described in equation (2-32).

Normally, the inputs to the TSK model are crisp numbers. Therefore, the matching degree of the input to the i^{th} rule is computed using the “min” operator.

$$\alpha_i = \min(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2), \dots, \mu_{A_{ir}}(x_r)) \quad (2-36)$$

or the product operator

$$\alpha_i = \mu_{A_{i1}}(x_1) \times \mu_{A_{i2}}(x_2) \times \dots \times \mu_{A_{ir}}(x_r) \quad (2-37)$$

The strength of the TSK model is its learning accuracy and expressiveness for a highly nonlinear functional relation using a small number of rules. The accurate global learning ability motivates the practical applications of the TSK model in nonlinear system estimation [Yen 1999]. The expressiveness provides a course of interpretability of the local models in the localized subspaces [Yen 1998], [Abonyi 2000], [Johansen 2000], [Johansen 2003]. Moreover, due to the explicit functional representation form, it is convenient to identify its parameters using some functional optimization algorithms; namely: the least-mean-squares method or the Kalman filtering. Some of the existing TSK-type of fuzzy systems are the ANFIS model developed by Jang [Jang 1992], [Jang 1993], and the Gan-Harris models [Gan 1999], [Harris 1999], [Gan 2001], [Harris 2001]. In principle, the locally weighted function can be an arbitrary-order polynomial function. However, it is usually adequate to present with a zero-order or a first-order function in practice. This is because the parameters in higher-order functions are more difficult to systematically identify and their corresponding degree of accuracy is seldom necessary. The TSK model is functionally similar to a single layer CMAC models [Sun 1996], local basis function expansion [Gan 1999], and locally weighted regression methods [Cleveland 1988]. In the latter case, the weighting of each data point to its local model is similar to the matching degree of that data point to its fuzzy subspaces in the TSK context.

Existing TSK fuzzy models proposed in the literature attempt to optimize the global learning accuracy as well as to maintain the interpretability of the local model. Most of the proposed methods suffer from the use of offline learning algorithms to globally optimize this multi-criteria problem [Yen 1998], [Johansen 2003]. Despite the ability to reach an optimal solution in terms of accuracy and interpretability, these offline methods are not applicable to learning in adaptive or incremental systems. Furthermore, most of these learning methods in the TSK

model are susceptible to the limitation of the curse-of-dimensionality. Section 3 of this Thesis will attempt to perform an in-depth study of these criteria in the design of the TSK model. These criteria are listed as follows; namely:

- the interpretability of the local model;
- the global accuracy; and
- the dimensionality of the system.

A generic TSK framework with online learning capability is then proposed to handle the different scenarios of this design problem in Section 3.

2.2.3 Fuzzy Reinforcement Learning Models

Besides the coupling of function approximation with reinforcement learning, fuzzy set theory has been employed in reinforcement learning as well. These fuzzy reinforcement learning models were demonstrated to effectively model continuous state and action reinforcement learning problems. These models are being surveyed in the following context.

Berenji first arrived at the idea of *fuzzy reinforcement learning* (FRL) [Berenji 1992]. The main idea of this actor-critic reinforcement learning model (*fuzzy temporal difference*) is using the temporal difference algorithm to tune the fuzzy logic controller's membership functions of the GARIC architecture [Berenji 1992]. The fuzzy actor-critic model with *Takagi-Sugeno fuzzy rule base* has been proven to converge in a later paper [Berenji 2001]. However, the proof is not exhaustive on continuous state-spaces, and is limited to the convergence of the actor component (not the critic component). Furthermore, the corresponding TSK fuzzy inference system is not well-defined. Fuzzy Q-learning [Berenji 1994] is defined as an extension to Bellman-Zadeh's fuzzy dynamic programming approach and Watkins's Q-learning. Firstly, the approximate and imprecise prior domain knowledge was expressed as fuzzy sets, and subsequently refined during the Q-learning fuzzy operations. Glorennec [Glorennec 1994] proposed a different version of fuzzy Q-learning; instead of using Q-learning for membership refinement, the method applied Q-learning to guide its rule selection. Fuzzy Q-learning has been shown to be extensible to continuous state and action spaces [Suh 1997]. Another model by Bonarini [Bonarini 1996], the *Evolutionary Learning of Fuzzy rules* (ELF), has been developed by employing the general idea of *Learning Classifier System* (LCS) in the learning of fuzzy logic controllers (FLC).

Berenji further combined Glorennec's fuzzy Q-learning and GARIC as the GARIC-Q [Berenji 1996], where multiple GARIC agents acting locally is controlled by a top level fuzzy Q-learning that selects one GARIC model with the best rule base. Glorennec introduced another fuzzy Q-learning [Glorennec 1997], where the Q-learning was used for online optimization of the consequence part of the fuzzy rules, the result of this model was compared against a genetic algorithm [Goldberg 1989] for the Cart-Centering Problem. *Fuzzy actor-critic learning* (FACL) and Fuzzy Q-learning were used in the conclusion part of the fuzzy system in Jouffe's paper [Jouffe 1998]; these techniques were experimented with the ventilation control task [Jouffe 1997]. In Kim's paper [Kim 1999], the *self-organizing fuzzy inference system based on Q-learning* (SOFIS-Q) was being proposed to resolve the continuous state and actions problems in Q-learning. Two reinforcement learning algorithms were used to design reinforcement learning system with the TSK fuzzy logic controllers in [Yan 2001]. They are the *Takagi-Sugeno fuzzy reinforcement learning* (TSFRL), its corresponding online tunable version using genetic algorithm, and the *Genetic Takagi-Sugeno fuzzy reinforcement learning* (GTSFRL). Deng [Deng 1997] has shown an integration of Q-learning, fuzzy logic and CMAC function approximator in the control of the pH-neutralization process.

Berenji extended the GARIC-Q to two similar versions of MULTI-GARIC-Q [Berenji 1998], where multiple agents can collaborate to investigate the state space within an extended GARIC-Q framework. Some benefits of having multiple fuzzy reinforcement learning agents have been shown in his subsequent papers [Berenji 1999], [Berenji 2000]. Berenji pointed out several strengths of the proposed models; namely: applicability in continuous state POMDPs; and having better performance for cooperative agents formulation. Another research group developed some hybrid systems with reinforcement learning and neuro-fuzzy approaches for learning and controlling linguistic information [Lin 1998]. Lin [Lin 1995] emphasized on using some extended version of actor-critic models to realize a control problem. The critic is formed by a neuro-fuzzy hybrid network to predict the value function, and the actor is formed by another neuro-fuzzy hybrid network to act as a stochastic action selector. Lin [Lin 1996a] employed a similar model (RFALCON) with a different underlying neuro-fuzzy hybrid network as the predictor.

Table 2-2: Survey of Fuzzy RL models

Fuzzy RL models	GARIC	FQL	FQL/DFQL	GARIC-Q
Fuzzifying	State-action space	Action-value	State-action space	State-action space
Fuzzy Model	Mamdani-liked	---	Mamdani-liked	Mamdani-liked
Eligibility Trace	0	0	0	0
RL Algorithm	Actor-Critic	Q-learning	Q-learning	Actor-Critic/Q-learning
Applications	Cart-pole	Simple navigation	Cart-pole	Cart-pole

Fuzzy RL models	Lin's models	FACL	FACL/FQL	SOFIS-Q
Fuzzifying	State-action space	State-action space	Action-value	State-action space
Fuzzy Model	Mamdani's model	Mamdani's model	Mamdani's model	Mamdani's model
Eligibility Trace	0	Accumulate	Accumulate	Replacing
RL Algorithm	Actor-Critic	Actor-Critic	Actor-Critic/Q-learning	Q-learning
Applications	Cart-pole/Ball-Beam	Ventilation control	Boat/Cart-pole/Mountain-Car	Cart-pole

Fuzzy RL models	GTSFRL	Deng's model	Suh's model
Fuzzifying	State-action space	State-action space	State-action space
Fuzzy Model	TSK model	TSK model	Fuzzy Interpolation
Eligibility Trace	Accumulate	No	No
RL Algorithm	Actor-Critic	Q-learning	Q-learning
Applications	Cart-pole	pH-neutralization	Simple 2-D task

These summarize the historical perspective on usages of fuzzy techniques in reinforcement learning. A simple survey of those architectures is presented in Table 2-2. The models in this study are GARIC [Berenji 1992], FQL [Berenji 1994], FQL/DFQL [Glorennec 1994], [Glorennec 1997], GARIC-Q [Berenji 1996], Lin’s models [Lin 1995], [Lin 1996a], [Lin 1998], FACL [Jouffe 1997], FACL/FQL [Jouffe 1998], SOFIS-Q [Kim 1999], GTSFRL [Yan 2001], Deng’s model [Deng 1997], and Suh’s model [Suh 1997].

The comparisons with respect to five different perspectives are listed below.

- Fuzzifying or fuzzy techniques are used in which part of the model. The modelling of continuous spaces reinforcement learning can be achieved by fuzzifying the state-spaces or state-action spaces. This is the most significant strength of applying fuzzy techniques in reinforcement learning. There are some techniques to fuzzify the action-value function or reinforcement signal, but such techniques are not justifiable for their practical usage.
- Fuzzy model: The fuzzy models adopted in reinforcement learning are generally the Mamdani and the TSK models. The fuzzy models are used as the function approximator in these RL models. However, the convergence results of most of these fuzzy reinforcement learning models are not studied.

- Eligibility traces: Eligibility trace formulation is able to extend the offline experience replay (or backup of return values) into an online learning paradigm. The models with eligibility trace of $\lambda=0$ are equivalent to asynchronous dynamic programming, where their learning are not as efficient as the general $TD(\lambda)$ learning for delayed reinforcement task. Furthermore, a replacing trace implementation is an unbiased estimation of the value function, which is better than an accumulating trace implementation [Singh 1996].
- RL algorithm: Fuzzy models act as the function approximators in the structural credit assignment of reinforcement learning systems. However, the temporal learning component can be substituted with any of the popular models; namely: actor-critic model, Q-learning and SARSA models.
- Application area: The application area serves as a platform to study the efficiency of the fuzzy reinforcement learning systems. Those models with more complex applications such as cart-pole and mountain-car problems are more convincing in practice.

By examining the survey, some deficiencies in these models are identified. Most existing fuzzy reinforcement learning models encountered one or more of the following problems. They are listed as follows; namely:

- Weakly defined roles, and are not directly positioned as the function approximator.
- Weakly defined eligibility trace formulation.
- Not generally applicable to other classes of temporal learning algorithms.

Although several fuzzy reinforcement learning models have been proposed, only few of these models focus on approximating the value function [Berenji 1992], [Glorennec 1994], [Berenji 1996], [Lin 1996a], [Jouffe 1998], [Kim 1999], and even fewer of these models [Lin 1996a], [Jouffe 1998] are explicitly positioned as the function approximator's role in the reinforcement learning paradigm. Thus, these models are not being explicitly analysed theoretically from the function approximation perspective of reinforcement learning. Subsequently, most of these fuzzy reinforcement learning models have weak design of eligibility trace formulation. Furthermore, these models assume a particular temporal learning algorithm, and may not be generally applicable to the general classes of temporal learning algorithms. In contrast, Van Roy [Van Roy 1998] has provided solid theoretical studies on the convergence properties of reinforcement learning based on the local basis function approximation. As will be described in Section 4.2, the TSK fuzzy models are a special case of local basis function approximation. This justifies the motivation of this Thesis in

formulating a TSK model for the value function approximation problem in reinforcement learning.

Section 4 of this Thesis proposes a modularly designed generic reinforcement learning framework to solve the above-mentioned deficiencies. The framework is demonstrated to be generally applicable to three popular temporal learning algorithms with this modular design. A fuzzy system is subsequently mapped to implement the role of the function approximator in this generic framework. The eligibility trace formulation in this framework is extended from the formulation of local basis function method, which is properly defined. The stability and convergence issues of this framework are discussed based on the convergence result of local basis function approximation. This provides a theoretically solid foundation to the generic reinforcement learning framework.

3 FITSK: The Generic Fuzzy Input Takagi-Sugeno-Kang Fuzzy Framework

*"The marble not yet carved can hold the form of every thought the greatest artist has."
~ Michelangelo Buonarroti*

The Takagi-Sugeno-Kang (TSK) fuzzy model [Sugeno 1988], [Takagi 1985] is a class of fuzzy models that assumes local model representation with local function dynamics at the consequent or rule-layer of the models. This is described in Section 2.2.2 of the Thesis. The idea of such a fuzzy model is to compute the output by performing fuzzy interpolations of simpler local functional models in the neighboring fuzzy partitions. The main advantage of the TSK-model over other classes of fuzzy models is its ability to model a system accurately, either globally or locally. The accurate global learning ability motivates the practical applications of TSK-model in nonlinear system estimation [Yen 1999]. The local learning ability provides a course of interpretability of the local models in the localized subspaces [Yen 1998], [Abonyi 2000], [Johansen 2000], [Johansen 2003].

Existing TSK-model learning algorithms proposed in the literature can be classified using two criteria. The first criterion is essentially the locality of learning. The second criterion is being an online or an offline learning algorithm. The locality of learning depends on the model's learning objective function, which is a minimization problem of the global or the local learning errors. The global parameter tuning considers the minimization of the global error of the model, such as the ANFIS's Kalman filter algorithm [Jang 1992]. This benefits the global accuracy of the system, but suffers from degradation in local interpretability. This degradation exhibits an erratic local behavior, which causes the local models to be difficult in interpretation [Yen 1998]. In contrast, the local parameter tuning takes the form of the minimization problem of the local learning errors. This ensures the local models to have accurate representations and improves on the local interpretability of the system. Examples of such local parameter tuning are the direct update of zero-order TSK model and the local learning algorithm in Yen's model [Yen 1998]. However, minimization of local learning errors without consideration of global behavior results in degradation of global learning accuracy.

An offline learning algorithm assumes that the data is presented in a batch form and can be repeatedly accessed. It is easier in guaranteeing the success in reaching an optimal solution for such form of learning, based on its learning objective function. Furthermore, it has the

flexibility in recalling the stored training examples to improve on the quality of learning. However, the offline learning is inapplicable to adaptive or incremental learning systems; such as the online reinforcement learning paradigm. Moreover, it is prone to the curse-of-dimensionality problem [Bellman 1957]. Examples of the offline learning methods includes the singular value decomposition approach [Yen 1998] and the constraint-optimization method [Johansen 2003]. Summarizing these, the existing TSK-models encountered one or more of the following major problems despite being good modeling tools for nonlinear system estimation. They are:

- offline learning;
- low interpretability of local models;
- degradation in global learning accuracy;
- inapplicable to higher dimensional system.

Hence, a novel TSK-based framework that is immune to the above deficiencies is proposed in this section. The framework is termed *the generic fuzzy input Takagi-Sugeno-Kang fuzzy framework* (FITSK). This section is organized as follows. Section 3.1 describes the general structure of the FITSK framework, the zero-order and the first-order FITSK models, and their online learning algorithms. Section 3.2 presents the benchmarking results of the FITSK models with five nonlinear system estimation tasks. Section 3.3 summarizes this section.

3.1 FITSK

This section introduces the generic FITSK framework; the FITSK framework takes the form of a superset of all the Takagi-Sugeno-Kang fuzzy models [Takagi 1985], [Sugeno 1988]. A special case of the FITSK framework is mapped to a zero-order FITSK model (denoted by FITSK^0) on the assumption that the local models' function dynamics are zero-order (constant term). Subsequently, a degree of localization is proposed to control the locality of learning. It is introduced in the zero-order FITSK model to balance the bias-variance of the learning process. A first-order FITSK model (denoted by FITSK^1) is another adaptation of the FITSK framework, where the local models are first-order linear functions of the input vector. There are two classes of parameter learning for the FITSK^1 model. They are the global Kalman filter algorithm [Jang 1992] and the localized Kalman filter algorithm that will be discussed in this section. The latter is computationally more effective than the former. A degree of localization is introduced in the localized Kalman filter algorithm to control the

interpretability of the FITSK¹ model. A guideline is proposed for the selection of different FITSK models with respect to the following three design criteria; namely:

- model interpretability;
- learning accuracy;
- data dimensionality.

The guideline allows the judicious selection of the FITSK models with different settings to generate a balanced system on the basis of these criteria.

3.1.1 Generic Structure of FITSK

Figure 3-1: Generic Structure of FITSK.

The FITSK framework consists of six layers of nodes. The six layers are the input fuzzifier (IF), the input linguistic (IL), the conjunction (C), the normalization (N), the functional or rule (F), and the summation (S) layers. The input fuzzifier nodes in the first layer fuzzify the input vector into a corresponding vector of fuzzy membership functions. Subsequently, the input linguistic layer measures the similarity of each fuzzy membership function with its corresponding linguistic nodes. The conjunction layer computes the conjunction of all incoming signals and generates the corresponding weights. The normalization layer normalizes these weights. Each node of the functional or rule layer implements a local model, which is a form of the TSK rule. The outputs of the functional layer are weighted with their incoming normalized weights. Finally, the summation layer summarizes all the incoming signals. Therefore, the outputs of the FITSK are computed by interpolating the local models with the normalized fuzzy measures of the input vector. This paragraph briefly summarizes the operations of the generic FITSK framework; subsequent segment of this subsection formulates these mathematically.

Each input fuzzifier node $\mathbf{IF}_i$, $i \in \{1, 2, \dots, I\}$, has a single input. Vector $X = [x_1, \dots, x_i, \dots, x_I]^T$ represents the inputs to the FITSK. Each output node S_n , where $n \in \{1, 2, \dots, N\}$, computes a single output denoted by y_n . Vector $Y = [y_1, \dots, y_n, \dots, y_N]^T$ denotes the outputs of the FITSK with respect to the input vector X . The indices i, j, k, l, m, n are used to refer to arbitrary nodes in layers 1, 2, 3, 4, 5, and 6, respectively.

The total number of nodes for layers 1, 3, 4, and 6 are denoted with variables I, K, L , and N respectively. Each input fuzzifier node $\mathbf{IF}_i$ may have a different number of input linguistic (IL) nodes J_i . Hence, the total number of nodes for layer 2 is $\sum_{i=1}^I J_i$. Layer 3 consists of conjunction nodes $\mathbf{C}_k$, where $k \in \{1, 2, \dots, K\}$, and $1 \leq K \leq \prod_{i=1}^I J_i$. Each layer 3 node $\mathbf{C}_k$ is directly connected to a layer 4 normalization node $\mathbf{N}_l$. Hence, the total number of nodes in layers 3 and 4 are identical, $K=L$. Each layer 4 node $\mathbf{N}_l$ is associated with M number of layer 5 functional nodes, $\mathbf{F}_{l,m}$. M is identical to the number of layer 6 nodes, $M=N$. Hence, the total number of nodes in layer 5 is $L \times M$. Each output node at layer 6 is a summation node, S_n . Each summation node is connected with L number of layer 5 functional nodes. The total number of nodes at layer 6, N , depends on the dimension of output vector Y . The FITSK model adopts the Takagi-Sugeno-Kang's fuzzy model and the trainable parameters are found in layer 5 of the model.

The output of a node is denoted as Z with the subscripts specifying its origin, for example Z_{IF_i} is the output of node IF_i . All the outputs of a layer are propagated to the inputs of the connecting nodes at the next layer with unity link-weight.

The generic operations of the FITSK can be defined as described by equations (3-1) - (3-9).

Layer 1 (Input fuzzifier layer):

$$Z_{IF_i} = \mu_i(x_i) \quad (3-1)$$

where

$\mu_i(\cdot)$ is the fuzzy membership function of input fuzzifier node IF_i .

Layer 1 nodes fuzzify a singleton input by adopting the input x_i as the center of the fuzzy membership function, such that the membership function achieves its maximum value at x_i and decreases while moving away from the center (equation (3-1)). However, when a non-singleton (fuzzy) input is presented, the node simply directs the non-singleton input as the output of the node.

Layer 2 (Input linguistic layer):

$$Z_{IL_{i,j}} = \mu_{i,j}(Z_{IF_i}) = \mu_{i,j}(\mu_i(x_i)) \quad (3-2)$$

where

$\mu_{i,j}(\cdot)$ is a fuzzy linguistic label of node IF_i .

Layer 2 nodes take in the fuzzy membership function of layer 1 outputs as the inputs, and compute a *fuzzy subethood measure* [Yen 1999] for each corresponding fuzzy linguistic label (equation (3-2)). The fuzzy subethood measure defines the degree that fuzzy set A is a subset of fuzzy set B as equation (3-3).

$$S(A, B) = \frac{|A \cap B|}{|A|} \quad (3-3)$$

If the *min* operator is used for intersection operation, then equation (3-3) is described in details in equation (3-4).

$$S(A, B) = \frac{\int_{x \in U} \min(\mu_A(x), \mu_B(x))}{\int_{x \in U} \mu_A(x)} \quad (3-4)$$

or can be approximated by equation (3-5).

$$S(A, B) = \frac{\max_{x \in U} (\min(\mu_A(x), \mu_B(x)))}{\max_{x \in U} (\mu_A(x))} \quad (3-5)$$

Layer 3 (Conjunction layer):

$$Z_{C_k} = \prod_{i=1}^I Z_{IL_{i,j}} \quad (3-6)$$

Layer 3 nodes compute the conjunction of all the incoming signals using a product or a minimum inference (equation (3-6)). Each input dimension contributes at most one incoming signals to a layer 3 node.

Layer 4 (Normalization layer):

$$Z_{N_l} = \frac{Z_{C_k}}{\sum_{k=1}^K Z_{C_k}} = w_l \quad (3-7)$$

where

w_l is the normalized weight.

Layer 4 nodes normalize the output from layer 3 nodes (equation (3-7)).

Layer 5 (Functional/Rule layer):

$$Z_{F_{l,m}} = Z_{N_l} f_{F_{l,m}}(X) = w_l f_{F_{l,m}}(X) \quad (3-8)$$

where

$f_{F_{l,m}}(X)$ is a local model function of node $F_{l,m}$.

Layer 5 nodes implement a product of the normalized weight with the local model function (equation (3-8)).

Layer 6 (Summation layer):

$$y_n = Z_{S_n} = \sum_{l=1}^L Z_{F_{l,m}} \quad (3-9)$$

Layer 6 nodes perform a summation operation on the corresponding layer 5 outputs (equation (3-9)).

Two motivations drive the development of the FITSK model. The first is to craft a systematic and generic framework for an arbitrary order TSK model [Takagi 1985], [Sugeno 1988] with arbitrary parameter learning capability. The second is to map the generic model with a non-singleton (fuzzy) input fuzzifier such that the *degree of localization* of the local model can be

controlled with ease. A special case of the generic FITSK is the ANFIS model [Jang 1992] which considers a singleton input fuzzifier at layer 1, a product inference scheme at layer 3, a first order polynomial local model function at layer 5, and a single output at layer 6.

3.1.2 A Zero-Order FITSK Model

The zero-order FITSK model is a special case of the generic FITSK framework with zero-order polynomial functions (constant terms) at layer 5, and product inference in layer 3. The specific formulation at layers 1 and 2 are described as follows:

Layer 1 (Input fuzzifier layer):

The choice of fuzzy membership function is flexible. In this formulation, a Gaussian membership function is chosen, as described in equation (3-10).

$$Z_{IF_i} = \mu_i(x_i) = e^{\frac{-(x-x_i)^2}{2\sigma_i^2}} \quad (3-10)$$

where

x spreads over the universe of possible range of x_i , and
 σ_i is the standard deviation of the Gaussian function.

The standard deviation, σ_i , reflects the width of the membership function, and is dependant on the accuracy and generalization to be achieved by a specific system. This is a *bias-variance dilemma* problem [Geman 1992]. In the ‘bias’ case, smaller value of the standard deviation implies that the learning of the local model is highly focus in the local subspace, and directly resulting in higher training accuracy of the system. Conversely, in the ‘variance’ case, a larger standard deviation anticipates higher noise or generalization capability in the neighbouring subspaces.

Layer 2 (Input linguistic layer):

Each linguistic node is a singleton label to simplify the computation of fuzzy subsethood measure. Let $s_{i,j}$ denote the centre of the singleton of the j^{th} linguistic label of the i^{th} input node, $\mathbf{IL}_{i,j}$, with unity height. The computations involving layer 1 and layer 2 nodes can be simplified as equation (3-11). Simplifying this will equivalently convert the zero-order FITSK model into a zero-order ANFIS model [Jang 1992]; this contributes to one of the specific form of the FITSK model.

$$Z_{\mathbf{IL}_{i,j}} = e^{\frac{-(s_{i,j}-x_i)^2}{2\sigma_i^2}} \quad (3-11)$$

The system is design to let the designer specifying the number of labels in each dimension. The location of $s_{i,j}$ is evenly distributed over the value range of the input dimension i . The resultant resolution, r_i , is computed as equation (3-12).

$$r_i = \frac{\text{range}_i}{\text{no. of labels}_i - 1} \quad (3-12)$$

where

range_i is the value range of the input dimension i , and
 no. of labels_i is the total number of labels in dimension i .

Layer 5 (Functional layer):

The local model function as discussed in equation (3-8) is implemented with a zero-order function or constant term in the FITSK⁰ model, as shown in equation (3-13).

$$\begin{aligned} f_{F_{l,m}}(X) &= c_{F_{l,m}} \\ Z_{F_{l,m}} &= w_l c_{F_{l,m}} \end{aligned} \quad (3-13)$$

where

$c_{F_{l,m}} \in \Re$ is the zero-order function (constant term) of the local model in the node $F_{l,m}$.

Let the vector $D = [d_1, \dots, d_n, \dots, d_N]^T$ denote the actual outputs of the FITSK with respect to the input vector X . The training assumes a direct adaptive learning process, which has more plasticity rather than stability. Higher plasticity means that the learning process adapts to new information rapidly by forgoing past information. Thus, the zero-order FITSK model is effective in modelling nonlinear adaptive systems such as time-series data or temporally adaptive systems. The learning process for an arbitrary zero-order local model is explained by equation (3-14).

$$\begin{aligned} \Delta c'_{F_{l,m}} &= \alpha w_l (d'_n - y'_n), \text{ and} \\ c'^{t+1}_{F_{l,m}} &= c'^t_{F_{l,m}} + \Delta c'_{F_{l,m}} \end{aligned} \quad (3-14)$$

where

t is the time index, and
 $\alpha \in (0, 1]$ is the positive step size learning constant.

If the stability of the learning system is crucial, the learning constant can be modified to a monotonically decreasing function to tighten the plasticity of the system.

The design of input fuzzifier with singleton linguistic formulation is driven by two motivations. The first is to exclude the need of online clustering process for the linguistic

labels. The second motivation is to ease the control of the *degree of localization* in the neighboring subspaces.

Clustering process for the Mamdani's fuzzy model [Mamdani 1975] is an unsupervised learning process by partitioning the input and output spaces into reasonable clusters [Bezdek 1981], [Kohonen 1982], [Tung 2002]. However, unsupervised clustering process for TSK fuzzy model identification is unjustifiable because the quality of the cluster is dependant on the input data as well as the corresponding output dynamics of the system. Several clustering techniques [Männle 1999], [Takagi 1985] have been proposed for TSK model identification in a supervised learning manner, in which the quality of the output is quantified with respect to the input clusters formation. However, they suffer from adopting a heuristic search approach and are relatively computational expensive [Männle 1999], [Takagi 1985]. This demonstrates that the supervised clustering problem for identification of TSK fuzzy model structure is a combinatorial complex problem. In order to reduce the computational cost, the FITSK model presupposes an even space partitioning of the singleton linguistic labels. This partitioning is presumed in FITSK model instead of adopting some computationally expensive online clustering technique. Normally, a TSK model maintains the learning accuracy by fuzzy interpolation of neighbouring local models. The FITSK is able to maintain a similar fuzzy interpolation by having fuzzy input with singleton linguistic labels, without the need of having a clustering process to structurally define the optimal linguistic labels. This is because the fuzzification is performed prior to the linguistic layer, and the membership functions are well-defined in the sense that the centers of the membership functions represent the exact location of the input vector.

The standard deviation in equation (3-11) is defined as a function of the input value range in equation (3-15).

$$\sigma_i = sr^{ld} \times \text{range}_i / 5 \quad (3-15)$$

where

$sr \in \mathfrak{R}$ is the shrinking rate and $sr \in (0,1]$,
 range_i is the value range of the input dimension i , and
 ld is the degree of localization.

The maximum of the standard deviation is estimated as $\text{range}_i/5$ such that the membership function is able to contain the entire range of the input space. This ensures that the learning error is systematically distributed across the entire learning space. The minimum of the standard deviation is chosen with respect to the concept of ε -completeness [Lee 1990a], [Lee

1990b]. The level ε at the crossover point of neighboring fuzzy membership functions implies a strong belief in the positive sense of the fuzzy control rules, which are associated with the controller. The degree of belief is chosen to be greater than 0.5 [Lee 1990a], [Lee 1990b] to achieve this. Thus, the weight or matching degree of each label is selected to be greater than 0.5.

From equation (3-11), this represents that

$$\begin{aligned}
 0.5 &\leq e^{\frac{-(s_{i,j}-x_i)^2}{2\sigma_i^2}} \\
 \Rightarrow \sigma_i &\geq 0.8493 \times (s_{i,j} - x_i) \\
 \Rightarrow \sigma_i &\geq 0.8493 \times \frac{r_i}{2} \\
 \Rightarrow \sigma_i &\geq 0.4247 r_i
 \end{aligned} \tag{3-16}$$

Therefore, the standard deviation is set to be at least 0.4247 times of the resolution in that dimension.

Equation (3-15) is able to control the bias-variance of the learning system by varying the degree of localization, ld . This is illustrated with a motorcycle crash dataset from [Hardle 1990]. The dataset consists of a one-dimensional time-series of 133 accelerometer readings in an experiment on the efficacy of crash helmets, as shown in Figure 3-2.

Figure 3-2: The motorcycle crash dataset.

Experiment was performed by randomly splitting the dataset into 67 training datasets and 66 testing datasets; setting the shrinking rate, $sr = 0.95$; and the number of labels to 20. Each set of the results was acquired by setting a degree of localization, ld , and with each model being trained for 200 iterations.

Figure 3-3: Training and testing mean-squared-errors of FITSK.

Figure 3-3 illustrates the training and testing results of this experiment. The degree of localization directly controls the width of the membership function. The training accuracy of the system is proportional to the width of the membership function. However, the testing error increases when the degree of localization is more than 16, representing a scenario of overfitting or overtraining. This demonstrates that it is possible to control the bias-variance of the learning system by simply modifying the degree of localization.

Thus, the bias-variance dilemma is a system dependant design issue. The FITSK model simplifies the design process by using the degree of localization to control the bias-variance. The degree of localization defines the locality or the vagueness of the input vector to neighboring subspaces. This locality represents how focus is an input to a local region. A higher degree of localization refers to a higher accuracy (bias) and less generalization (variance), and vice-versa. The locality of local models can be controlled by varying the degree of localization of the FITSK model without the need of repositioning or redefining the linguistic labels.

A possible usage of varying the degree of localization is to relate it to the progress of training in an adaptive system, such that the degree of localization is gradually increased as learning progresses. This ensures that the generalization capability of the learning system is

guaranteed when learning commences, and gradually moves towards a higher learning accuracy.

3.1.3 A First-Order FITSK Model

The FITSK framework takes the form of a first order FITSK model by adopting first order local models at layer 5 (functional layer) of the framework. This is achieved by substituting a first order linear function into equation (3-8) of the generic FITSK framework, giving rise to equation (3-17).

$$\begin{aligned} Z_{F_{l,m}} &= w_l f_{F_{l,m}}(X) \\ f_{F_{l,m}}(X) &= b_{0,(l,m)} + b_{1,(l,m)}x_1 + \dots + b_{i,(l,m)}x_i + \dots + b_{I,(l,m)}x_I \end{aligned} \quad (3-17)$$

where $B_{(l,m)} = [b_{0,(l,m)}, b_{1,(l,m)}, \dots, b_{i,(l,m)}, \dots, b_{I,(l,m)}]^T$ refers to the parameter vector of an arbitrary node, $F_{l,m}$, in layer 5 of the FITSK. The inference performed by the FITSK model is an interpolation of all the relevant local linear models based on the relevance of the input data to the fuzzy subspaces associated with the linear model. The output nodes at layer 6 take the form of equation (3-18).

$$y_n = \sum_{l=1}^L w_l f_{F_{l,m}}(X) = \sum_{l=1}^L w_l (b_{0,(l,m)} + b_{1,(l,m)}x_1 + \dots + b_{i,(l,m)}x_i + \dots + b_{I,(l,m)}x_I) \quad (3-18)$$

For simplicity of discussion, assume that an FITSK model has two inputs x_1 and x_2 , and a single output y , with only two rules, then

$$y = \sum_{l=1}^2 w_l f_{l,1}(x_1, x_2) = w_1 (b_{0,(1,1)} + b_{1,(1,1)}x_1 + b_{2,(1,1)}x_2) + w_2 (b_{0,(2,1)} + b_{1,(2,1)}x_1 + b_{2,(2,1)}x_2) \quad (3-19)$$

Consider that the learning system is presented with P training examples, $(X^{(1)}, d^{(1)})$, $(X^{(2)}, d^{(2)})$, ..., $(X^{(p)}, d^{(p)})$, ..., $(X^{(P)}, d^{(P)})$, where

p is the index,

$X^{(p)}$ is the p^{th} input vector, $p \in \{1, 2, \dots, P\}$, and

$d^{(p)}$ is the p^{th} desired output value.

Equation (3-19) can be arranged in a matrix form described in equations (3-20) or (3-21).

$$AB = C \quad (3-20)$$

or

$$\begin{bmatrix} w_1^{(1)} & w_1^{(1)}x_1^{(1)} & w_1^{(1)}x_2^{(1)} & w_2^{(1)} & w_2^{(1)}x_1^{(1)} & w_2^{(1)}x_2^{(1)} \\ w_1^{(2)} & w_1^{(2)}x_1^{(2)} & w_1^{(2)}x_2^{(2)} & w_2^{(2)} & w_2^{(2)}x_1^{(2)} & w_2^{(2)}x_2^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_1^{(P)} & w_1^{(P)}x_1^{(P)} & w_1^{(P)}x_2^{(P)} & w_2^{(P)} & w_2^{(P)}x_1^{(P)} & w_2^{(P)}x_2^{(P)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_1^{(P)} & w_1^{(P)}x_1^{(P)} & w_1^{(P)}x_2^{(P)} & w_2^{(P)} & w_2^{(P)}x_1^{(P)} & w_2^{(P)}x_2^{(P)} \end{bmatrix} \begin{bmatrix} b_{0,(1,1)} \\ b_{1,(1,1)} \\ b_{2,(1,1)} \\ b_{0,(2,1)} \\ b_{1,(2,1)} \\ b_{2,(2,1)} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \vdots \\ d^{(P)} \\ \vdots \\ d^{(P)} \end{bmatrix} \quad (3-21)$$

with dimensions of matrices A , B , and C being $P \times L$, $L \times 1$, and $P \times 1$ respectively, where L is the total number of linear parameters.

Since the number of training examples is usually greater than the number of parameters to be identified, equation (3-21) is an over-determined problem and there is no exact solution. However, online estimation of the solution is possible with the formulation of recursive-least-squares algorithm (a special case of the Kalman filter algorithm) in order to perform iterative linear least square estimation (LSE), as described in [Jang 1992] and formally listed in [Lin 1996b].

Let a_p be the p^{th} row vector of matrix A , then B can be iteratively estimated using equation (3-22).

$$\begin{aligned} B_{p+1} &= B_p + S_{p+1} a_{p+1}^T \overbrace{(d^{(p+1)} - a_{p+1} B_p)}^{\text{Global learning error}}, \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{1 + a_{p+1} S_p a_{p+1}^T}, \quad p = 0, 1, \dots, P-1 \end{aligned} \quad (3-22)$$

with initial conditions of

$$B_0 = \mathbf{0} \quad \text{and} \quad S_0 = \gamma \mathbf{I},$$

where γ is a large positive number and $\mathbf{I}$ is the identity matrix of dimension $L \times L$. $\gamma \geq 10^6$ is usually adequate in the actual implementation of this algorithm.

The strengths of using the Kalman filter algorithm for the first order FITSK parameter identification are its online learning capability, fast convergence speed with minimal learning iterations, and optimal global learning performance [Xu 2002], [Yen 1998], [Johansen 2000]. However, it encounters two major problems. Firstly, the recursive-least-squares algorithm has high computational and space complexities. For example, a first order TSK fuzzy model of I -input, L -rules will have an S matrix of dimension $L^2(I+1)^2$, and the computational cost is in the order of $O(L^2(I+1)^2)$. Secondly, the system identified using the algorithm has erratic local behaviour and the local models have low local interpretability [Yen 1998], [Johansen 2000].

This jeopardizes the initial motivation of using the TSK model to gain insights into the local models, such that interpretability of the TSK model is possible [Yen 1998]. Therefore, the learning algorithm of the TSK model is a multi-objective identification problem that would have good global learning accuracy and local model interpretability [Johansen 2000].

A simple experiment was performed to demonstrate the exponential growth in computational complexity with FITSK when adopting the Kalman filter parameter learning (described in equation (3-22)). Consider a multi-dimensional function of the form described by equation (3-23).

$$y = \sum_{i=1}^{Dim} x_i^2 \tag{3-23}$$

where
Dim is the dimension of the function.

Figure 3-4: Time-complexity of FITSK with Kalman filter learning.

Figure 3-4 illustrates the time-complexity of the first order FITSK with the Kalman filter algorithm, which is used for parameter learning of the model. It is simulated to model function (3-23) in a series of input dimensions. There are 3 labels for each dimension at layer 2 of FITSK in this experiment. The time-complexity of the learning process suffers from an exponential order as the dimension increases. This demonstrates the seriousness of the curse-of-dimensionality problem [Bellman 1957] in the FITSK model with the global Kalman filter learning. Similar curse-of-dimensionality issue can be easily replicated in any general TSK model with a global learning algorithm; for example, the ANFIS model [Jang 1992], [Jang 1993].

With the hindsight from the above discussion, the modeling of the first order FITSK should be positioned as a multi-objective problem; namely:

- relaxing the computational complexity;
- improving the local model interpretability;
- maintaining the global learning accuracy.

These motivate the derivation of a localized version of Kalman filter, which is able to achieve these three design objectives, and concurrently being an online algorithm with fast convergence performance.

Consider a local approximation of an arbitrary local model (denoted by (l,m)) in layer 5 of FITSK. Each training example is being influenced by the weight w_l . Thus, the local approximation can be represented as input-output relationships as shown in equation (3-24).

$$\begin{bmatrix} w_l^{(1)} & w_l^{(1)}x_1^{(1)} & w_l^{(1)}x_2^{(1)} \\ w_l^{(2)} & w_l^{(2)}x_1^{(2)} & w_l^{(2)}x_2^{(2)} \\ \vdots & \vdots & \vdots \\ w_l^{(p)} & w_l^{(p)}x_1^{(p)} & w_l^{(p)}x_2^{(p)} \\ \vdots & \vdots & \vdots \\ w_l^{(P)} & w_l^{(P)}x_1^{(P)} & w_l^{(P)}x_2^{(P)} \end{bmatrix} \begin{bmatrix} b_{0,(l,m)} \\ b_{1,(l,m)} \\ b_{2,(l,m)} \end{bmatrix} = \begin{bmatrix} w_l^{(1)}d^{(1)} \\ w_l^{(2)}d^{(2)} \\ \vdots \\ w_l^{(p)}d^{(p)} \\ \vdots \\ w_l^{(P)}d^{(P)} \end{bmatrix} \quad (3-24)$$

in a form of that can be identified by the least-squares algorithm, $A_{(l,m)}B_{(l,m)} = C_{(l,m)}$.

To simplify the discussion, the subscript (l,m) is omitted since each local model is updated locally. Letting a_p be the p^{th} row vector of matrix A , and

$$\begin{aligned} \text{Given, } a_p &= [1 \quad x_1^{(p)} \quad x_2^{(p)}], \text{ then} \\ a_p' &= [w^{(p)} \quad w^{(p)}x_1^{(p)} \quad w^{(p)}x_2^{(p)}] = w^{(p)}[1 \quad x_1^{(p)} \quad x_2^{(p)}] = w^{(p)}a_p \end{aligned} \quad (3-25)$$

then B can be iteratively estimated by substituting the solution from equation (3-22) with equation (3-25), and is described in equation (3-26).

$$\begin{aligned} B_{p+1} &= B_p + (w^{(p)})^2 S_{p+1} a_{p+1}^T \overbrace{(d^{(p+1)} - a_{p+1} B_p)}^{\text{Local learning error}}, \\ S_{p+1} &= S_p - \frac{(w^{(p)})^2 S_p a_{p+1}^T a_{p+1} S_p}{1 + (w^{(p)})^2 a_{p+1} S_p a_{p+1}^T}, \quad p = 0, 1, \dots, P-1 \end{aligned} \quad (3-26)$$

and the local learning error is shown in equation (3-27),

$$\delta_{p+1} = (d^{(p+1)} - a_{p+1} B_p) \quad (3-27)$$

which is an approximation of the global learning error as shown in equation (3-28) (see also equation (3-22)),

$$\delta_{p+1} = (d^{(p+1)} - y^{(p+1)}) \quad (3-28)$$

Therefore, equation (3-26) can be simplified into equation (3-29).

$$\begin{aligned} B_{p+1} &= B_p + (w^{(p)})^2 S_{p+1} a_{p+1}^T \delta_{p+1}, \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{\frac{1}{(w^{(p)})^2} + a_{p+1} S_p a_{p+1}^T}, \quad p = 0, 1, \dots, P-1 \end{aligned} \quad (3-29)$$

which is the optimal recursive solution to the least-squares problem $AB=C$. Therefore, the mean-squared-error (MSE) criterion is the optimal objective function for this least-squares-minimization problem. Subsequently, the optimal locally weighted mean-squared error criterion of a local model is computed with equation (3-30).

$$Local\_MSE = \frac{1}{P} \sum_{p=1}^P (w^{(p)} \delta_p)^2 \quad (3-30)$$

The local_MSE criterion is the objective measure of the learning accuracy of a locally weighted local model. The total locally weighted mean-squared error of all the local models is the sum of all local_MSEs.

$$Local\_MSE_{total} = \frac{1}{P} \sum_{l=1}^L \sum_{m=1}^M \sum_{p=1}^P (w_l^{(p)} \delta_{(l,m),p})^2 \quad (3-31)$$

where

$L \times M$ is the total number of local models (from Section 3.1.1), and

(l,m) is the indexing of the local model.

On the other hand, equation (3-29) can be generalized using a degree of localization, ld .

$$\begin{aligned} B_{p+1} &= B_p + (w^{(p)})^{ld} S_{p+1} a_{p+1}^T \delta_{p+1}, \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{\frac{1}{(w^{(p)})^{ld}} + a_{p+1} S_p a_{p+1}^T}, \quad p = 0, 1, \dots, P-1 \end{aligned} \quad (3-32)$$

with initial conditions of

$$B_0 = \mathbf{0} \quad \text{and} \quad S_0 = \gamma \mathbf{I},$$

and a degree of localization of value 2 will always result in the optimal error as shown in equation (3-30).

Equation (3-32) is a proposed, localized version of Kalman filter or recursive-least-squares algorithm. Each local model has its local copy of S matrix and B vector. The parameter tuning

is computed locally instead of globally with respect to all the local models. The influence of an individual local model for a training example is measured with its weight. The degree of localization indicates the locality of the learning to the local model, and is indirectly proportional to the influence of training examples that are further away in the input spaces to a particular local model. A higher degree of localization will often result in better interpretability of local models.

Consider that K data points nearest to the centre of a local model in the local subspaces are responsible for influencing the interpretability of the local model. Then, the interpretability of the local models is measured with a K -central-nearest MSE (KCN_MSE) criterion, which is computed by equation (3-33).

$$KCN\_MSE = \frac{1}{N \times K} \sum_{n=1}^N \sum_{k=1}^K \delta_{n,k}^2 \quad (3-33)$$

where

$\delta_{n,k}$	is the error of the n^{th} local model w.r.t. the k^{th} data point,
K	is the total number of data points under consideration,
k	is the index, and
N	is the total number of FITSK's outputs, indexes by n .

The level of interpretability of local models is related to the degree of localization, and will be demonstrated with a benchmarking problem in Section 3.2.5 using the criterion described in equation (3-33).

The first order FITSK fuzzy model with localized Kalman filter of I -input, L -rules will have S matrices of dimension $(I+1)^2$. The localized Kalman filter is repeated for each local model during the training process, the computational cost is in the order of $O(L(I+1)^2)$. Furthermore, the weight (equation (3-7)) of most local models is zero or near-to-zero for high-dimensional FITSK models during a particular training instance. This is because only the weights of local models that are nearest to the input data in the multi-dimensional spaces are activated. The computational cost can be further reduced by suppressing the training of these near-to-zero weight local models. This significantly reduces the computational and space complexities, especially for higher-dimension FITSK model. This is possible for a particular training instance, when the support of the membership function spread across a relatively small region compare with the whole training region to model.

Figure 3-5: Time-complexity of FITSK with localized Kalman filter learning.

Figure 3-5 illustrates the time-complexity of a first order FITSK with the proposed localized Kalman filter algorithm for parameter learning of the function described by equation (3-23) with a different number of input dimensions. There are 3 labels for each dimension at layer 2 of FITSK in this experiment. The time-complexity of the learning process is still of an exponential order. However, the computational cost has been significantly reduced by comparing with the time-complexity of FITSK with global Kalman filter as shown in Figure 3-4. By comparing Figure 3-5 and Figure 3-4, the computational cost for a 10-dimensional data with localized Kalman algorithm is lower than the cost of the global Kalman algorithm with 5-dimensional data. For a 6-dimensional problem, the time complexity of the global Kalman filter algorithm is $2^{9.17}$, in contrast to the localized version, which is $2^{-3.0}$. This represents a 4615 times reduction in computational complexity. Moreover, for dimensions larger than 6, the problem becomes empirically intractable with the global Kalman filter algorithm. However, the localized version can be successfully applied up to 10-dimensional problems with less difficulty. This demonstrates that the FITSK model adopting the localized Kalman filter algorithm has superior immunity against the curse-of-dimensionality issue.

This research focuses on providing a general FITSK framework such that a specific model can be formulated with ease depending on different design criteria and considerations. The criteria for considerations are the interpretability, accuracy and dimensionality of the system; all these criteria are design or problem-dependant. Formulating this design problem as a multi-objective optimization problem is not justifiable practically since these objectives are system-dependant. Therefore, a guideline is provided to cater for the selection of FITSK model based on these three criteria.

Figure 3-6: Flowchart on 3-criteria model selection of FITSK.

Figure 3-6 illustrates the design guideline on the model selection of FITSK based on three criteria. If the interpretability of a learning system is of the highest priority, a first-order FITSK model with localized version of Kalman filter is adopted with a high degree of localization. This is to ensure that the learning is performed locally for each regional subspace. However, a zero-order FITSK model is chosen when both interpretability and accuracy is not important, and a straightforward and fast adaptive system is required. Finally, a first-order FITSK model is necessary on the condition that accuracy remains an important criterion. The global Kalman filter learning algorithm is adopted when the system has a low dimensionality since this approach is computationally costly. On the other hand, for higher dimensional systems, the localized Kalman filter is chosen. Since interpretability is not a main consideration, a low degree of localization will adequately accommodate such need.

3.2 Benchmarking Results and Analysis

Five nonlinear estimation tasks are studied in this subsection; the first three benchmarking tasks are taken from [Nakanishi 1993], and are benchmarked against five models as listed in that paper. The forth task is taken from [Jang 1992], and is compared against the ANFIS model and TSK⁰-FCMAC model as proposed by [Ting 2002]. The last task is taken from

[Hardle 1990], to comprehend the interpretability issue [Yen 1998] of the first-order FITSK model. The brief descriptions of the five tasks are as follows:

- A nonlinear system.
- Human operation of a chemical plant.
- Daily price of a stock in a stock market.
- Mackey-Glass time series prediction.
- Motorcycle crashing problem.

The detail descriptions of each task are to be referred to each respective research [Nakanishi 1993], [Jang 1992], [Hardle 1990].

The specifications of these tasks are shown in Table 3-1.

Table 3-1: Specifications of tasks

Task	No. of dimensions	Selected dimensions	Train datasets	Test datasets
(a)	4	2	25	25
(b)	5	2	35	35
(c)	10	3	50	50
(d)	4	4	500	500
(e)	1	1	133	---

The third column of Table 3-1 shows the selected dimensions of the original problem via structure identification [Nakanishi 1993]. The mean-squared error and Pearson correlation coefficient are used to benchmark the results, The mean-squared error measures the performance accuracy in terms of training or testing result. The Pearson product moment correlation coefficient is a dimensionless index that ranges from -1.0 to 1.0 and reflects the extent of a linear relationship between two sequences of data.

Defining mean-squared error and root-mean-squared error between two data vectors as equation (3-34).

$$\begin{aligned}MSE(\vec{x}, \vec{y}) &= \frac{1}{n} (\vec{x} - \vec{y})^T (\vec{x} - \vec{y}) \\ RMSE(\vec{x}, \vec{y}) &= \sqrt{MSE(\vec{x}, \vec{y})}\end{aligned}\tag{3-34}$$

where,

- MSE is the mean-squared error function,
- $RMSE$ is the root-mean-squared error function,
- $\vec{x}, \vec{y}$ are two data vectors, and

n is the number of elements in the vector.

The Pearson correlation coefficient is defined in equation (3-35).

$$R(\vec{x}, \vec{y}) = \frac{C(\vec{x}, \vec{y})}{\sqrt{C(\vec{x}, \vec{x})C(\vec{y}, \vec{y})}} \quad (3-35)$$

where,

R is the Pearson correlation coefficient function,

$\vec{x}, \vec{y}$ are two sequence of data vectors, and

$C(.)$ represents the covariance between two data vectors.

The two data vectors for performance evaluations are the desired and the actual outputs from the model respectively. The FITSK⁰ model refers to the zero-order FITSK model with a learning rate of 0.2, and a standard deviation of 0.4247 times of the resolution. The FITSK¹-GKF denotes the first-order FITSK model with global Kalman filter learning (based on equation (3-22)), and FITSK¹-LKF represents the first-order FITSK model with localized version of Kalman filter learning (based on equation (3-32)).

3.2.1 Task (a): Nonlinear system

The nonlinear system was described in [Nakanishi 1993], the result of the FITSK⁰ model is compared against five other models, the Sugeno's P&P-G, P, P-G, Mamdani's model, and Turksen's IVCRI model [Nakanishi 1993]. These models differ in terms of modelling and theoretical basis, the details can be found in [Nakanishi 1993]. Turksen's PVAAR and IVAAR models are not included in the evaluation due to the large number of missing values in their results. This is because the performance results of these two models will significantly degrade by considering such missing values to be out-of-range, and are substituted with a default value of zero.

The training datasets and testing datasets are identical for all the models as shown in Table 3-1. The task is reduced to a two inputs ($X1$ and $X2$) single output problem. The FITSK⁰ structure adopts two labels at layer 2 of each dimension. The training of FITSK⁰ converged in 4 training epochs. The output of the model is plotted against the desired output of the testing datasets as shown in Figure 3-7.

Figure 3-7: Testing result of FITSK⁰ for task (a).

Table 3-2: Performance comparison for task (a).

Model	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
MSE	0.345	0.773	0.467	0.862	0.706	0.334
R	0.829	0.558	0.845	0.490	0.609	0.828

Table 3-2 shows the performance comparison for different models using the testing dataset. The FITSK⁰ model has the highest accuracy in terms of mean-squared error, and is among the best in terms of correlation of actual output with respect to the desired output of the test dataset.

3.2.2 Task (b): Human control of chemical plant

The input dimensions selected for this task are the first and the third dimension ($X1$ and $X3$), as shown in [Nakanishi 1993]. FITSK⁰ assumes a structure of two labels in layer 2 for each dimension. The testing result is taken after the 30th training epoch.

Table 3-3: Performance comparison for task (b).

Model	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
MSE	288810	642020	1930700	657970	258140	37359
RMSE	537.410	801.260	1389.500	811.150	508.070	193.280
R	0.980	0.938	0.990	0.937	0.993	0.997

Figure 3-8: Testing result of FITSK⁰ for task (b).

Figure 3-8 plots the FITSK⁰ testing result with respect to the desired testing dataset. It shows that the FITSK⁰ outputs are highly correlated with the desired outputs. This is supported by the numerical performance results as shown in Table 3-3. The table shows that the testing mean-squared error for the FITSK⁰ model is the lowest among all the models, which is 591% better than the next best model, Turksen's IVCRI model. The testing result of FITSK⁰ is better than other models in terms of root-mean-squared error. Furthermore, the FITSK⁰ model produces a correlation of near to one, which means that the FITSK⁰ model simulates a highly similar trend as the desired output.

3.2.3 Task (c): Daily price of a stock in a stock market

The input dimensions selected for this task are the 4th, 5th, and 8th dimensions (X_4 , X_5 and X_8), as shown in [Nakanishi 1993]. FITSK⁰ assumes a structure of three labels for each dimension in layer 2, resulting in $3^3=27$ rule nodes or local models. The testing result is taken after the 10th training epoch.

Figure 3-9: Testing result of FITSK⁰ for task (c).

Table 3-4: Performance comparison for task (c).

Model	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
MSE	94.232	35.037	168.903	40.839	93.022	33.513
R	0.706	0.886	0.700	0.865	0.661	0.884

Figure 3-9 plots the daily prices of a stock and its corresponding FITSK⁰ approximation. The result is quite encouraging in terms of monitoring the trend of the stock. The FITSK⁰ model has the lowest error in terms of mean-squared error, and is among the best in terms of correlation of output with respect to the desired output of the test dataset.

Table 3-5: Relative MSE for three tasks.

MSE	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
Task a	0.40	0.90	0.54	1.00	0.82	0.39
Task b	0.15	0.33	1.00	0.34	0.13	0.02
Task c	0.56	0.21	1.00	0.24	0.55	0.20

Table 3-6: Relative Pearson correlation coefficient for three tasks.

R	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
Task a	0.98	0.66	1.00	0.58	0.72	0.98
Task b	0.98	0.94	0.99	0.94	1.00	1.00
Task c	0.80	1.00	0.79	0.98	0.75	1.00

Table 3-5 summarizes the performances of models in the three tasks with MSE. The table normalizes each MSE with the best result within the row. Summarizing all the three tasks as

discussed above, FITSK⁰ is able to achieve the lowest mean-squared error for all the tasks with significant accuracy improvements over all the other models. Table 3-6 summarizes the performances of models in the three tasks with Pearson correlation coefficient. The table normalizes each coefficient with the best result within the row. As shown in the table, the FITSK⁰ model is the only model that is capable of achieving the optimal performances for all the three tasks. Models such as Sugeno's (P), (P-G) or Turksen's (IVCRI) performed well for a particular task but not the others. These nonlinear system estimation benchmarks demonstrate that the FITSK⁰ model outperforms these popular models despite being a simple zero-order local model TSK network.

3.2.4 Task (d): Mackey-Glass time series prediction

The dynamics of Mackey-Glass differential delay equation is defined in equation (3-36).

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \quad (3-36)$$

This time series is a popular benchmark problem considered by several researchers [Jang 1992], [Ting 2002].

The time series is obtained similarly as in Jang's thesis [Jang 1992]. The fourth-order Runge-Kutta method was applied to compute the numerical approximation for the time period of $0 \leq t \leq 2000$; with a time step of 0.1, initial condition of $x(0) = 1.2$, $\tau = 17$, and $x(t) = 0$ for $t < 0$.

From the computed series, 1000 input-output data pairs from $118 \leq t \leq 1117$ were extracted with the following format:

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)]$$

The goal of the task is to use known values of the time series of past 18th, 12th, 6th time steps, and the current time to predict the 6th instance ahead in time. The first 500 data pairs are used as the training dataset while the remaining 500 data pairs are used as the testing dataset.

The models to be studied are TSK⁰-FCMAC [Ting 2002], ANFIS [Jang 1992], FITSK⁰, FITSK¹-GKF and FITSK¹-LKF. FITSK¹-LKF implements the first-order FITSK model with the proposed localized Kalman filter approach with a localization factor of 9. The models to be compared consist of TSK-type fuzzy models of zero-order (TSK⁰-FCMAC, FITSK⁰), and first-order (ANFIS, FITSK¹-GKF, FITSK¹-LKF).

The training dataset was randomly permuted into 100 different training dataset sequences in TSK⁰-FCMAC benchmarking, while the other models had 3 different permutations of training dataset. All the models were executed on the same computer platform, with the debug version of executables being generated from C++ source codes (C for ANFIS).

Table 3-7: Testing results of Mackey-Glass series.

Model	No. of labels (layer 2)	Average RMSE	Std. dev. (RMSE)	Training epochs	Avg training time (s)	No. of rules
TSK ⁰ -FCMAC	4	0.02229	0.00468	20	1.92	218
FITSK ⁰	3	0.01842	0.00004	20	0.99	81
ANFIS	2	0.00295	0.00000	1	0.27	16
FITSK ¹ -GKF	2	0.00321	0.00000	1	0.33	16
FITSK ¹ -GKF	3	0.00139	0.00000	1	9.78	81
FITSK ¹ -LKF	2	0.03070	0.00000	1	0.09	16
FITSK ¹ -LKF	3	0.01926	0.00000	1	0.13	81

Table 3-1 lists the testing results of Mackey-Glass series for the various models. The training process lasted for 20 epochs for the zero-order models. The second column of the table lists the number of labels of the respective models for each input dimension at layer 2 of the models; and the number of rules of the models are computed as (no. of fuzzy labels)⁴ since there are four input dimensions. TSK⁰-FCMAC had 218 rules as those zero-weight rules were not included in the computation of the total number of rules.

Comparing the zero-order models, FITSK⁰ is able to achieve a lower testing error despite using a lesser number of rules than the TSK⁰-FCMAC model, which is due to the Gaussian membership function adopted in layer 2 of the model. The average training time of FITSK⁰ is acceptably low despite the usage of Gaussian membership function, which is computational costly comparing with the trapezoidal or triangular fuzzy functions in TSK⁰-FCMAC.

Next, it is appropriate to compare the two first-order models with two linguistic labels, the ANFIS model at the 3rd row and FITSK¹-GKF model at the 4th row. It is shown that both models have similar accuracy and training time since they both adopt the global Kalman filter learning algorithm. The minor differences may be due to programming differences. The 5th row demonstrates the FITSK¹-GKF model with three linguistic labels at layer 2, and thus a larger number of rules (local models). As expected, the accuracy improves significantly over the previous 2 models as the number of local models increases. However, the drawback of this is the significant growth in computational cost, which increases from 0.33s to 9.78s.

The next two models are FITSK¹-LKF with two and three linguistic labels in each dimension; both adopting localized Kalman filter parameter learning with the degree of localization being set to numeric value 2. The computational time of these two models demonstrates a significant reduction comparing with models with the global Kalman filter approach. On the other hand, the growth in computational cost due to the increased number of local models is insignificant (from 0.09s to 0.13s). This clarifies that the parameter learning algorithm with the localized Kalman filter approach has better resistance to the curse-of-dimensionality than the global Kalman filter approach. Moreover, by using the Kalman filter parameter learning approaches, all the first-order models are able to achieve fast learning convergence within a single training epoch. This inherits the online learning capability from the Kalman filter parameter identification technique.

The results presented in Table 3-7 mainly focus on the comparison of FITSK over other TSK-based system. The result of wavelet packet neural network [Wang 2001] is being compared as a reference of FITSK to the other state-of-the-art MLP-based network. The best result among the MLP-based network is a WP-MLP with hierarchical tree clustering model [Wang 2001] with network structure of (14:2:8). Its corresponding average testing MSE is 2.75×10^{-6} (or RMSE = 0.00166) with standard deviation of 3.24×10^{-7} . This result confirms that the best accuracy of FITSK¹ (RMSE = 0.00139) is similar to a state-of-the-art MLP-based model. Moreover, the learning of a first-order FITSK model is online and deterministic, and is inheriting the interpretability capability of the TSK-based model. However, these strengths are not necessarily present in an MLP-based model.

3.2.5 Task (e): Motorcycle crashing problem

The final task is the motorcycle crashing problem data taken from [Hardle 1990], which has been briefly introduced in Section 3.1.2. The data is revisited here to illustrate the interpretability issue in the first-order TSK model. The same dataset has been used in [Yen 1998] to study the interpretability issue of TSK fuzzy models.

Firstly, eight linguistic labels in layer 2 of the FITSK model are equally positioned across the space of range [2.4 57.6]. The standard deviation for the Gaussian membership function is chosen to be the minimum, which is 0.4247 of the resolution, as shown in equation (3-16). The global mean-squared error of the system identified is 459.126 by using the global Kalman filter parameter identification (equation (3-22)). The global learning error is slightly better than the one identified by Yen's model [Yen 1998], which is 460.62. Yen adopted a singular value decomposition method to identify redundant fuzzy partitions using a firing

strength matrix [Yen 1998]. However, the method is a form of unsupervised clustering; no output value or output performance indication is used to influence the result of clustering. Thus, the method can only identify the redundant fuzzy partitions among all the fuzzy partitions, but can not recognize the relevancy of the fuzzy partitions with respect to the output of the system. It can be examined from the result in [Yen 1998] that the fuzzy partitions identified by Yen's model is non-optimal; i.e., there should be a fuzzy partition at a time near-to 26ms since the output around this region changes drastically. However, the method is unable to monitor such drastic transition at the output space. Thus, Yen's fuzzy partition method performed no better than an equally spaced partition with a fuzzy input formulation as in the FITSK model. This is supported by the higher learning accuracy of the FITSK model than the one by Yen's model.

Figure 3-10: Global and local learning results of first-order FITSK models.

The interpretability of the first-order FITSK local models is investigated in Figure 3-10. Figure 3-10 illustrates the learning result of FITSK¹ with different settings. The original dataset is represented using black dots in the figure; the learnt-models are plotted with solid curves; and the corresponding eight local models are indicated by straight bold lines in the

figure and denoted numerically from (1) to (8). Figure 3-10(a) represents the training result of FITSK¹-GKF, which is the first-order FITSK model with global Kalman filter as the parameter learning algorithm. Figure 3-10(b), (c), and (d) are first-order FITSK models with localized Kalman filter (FITSK¹-LKF), the degrees of localization of these models are 2, 6, and 10 respectively. By observation, it is easy to infer that the learnt systems have significant differences in the 1st, 2nd and 4th local models given different system settings. The global learning in Figure 3-10(a) has low global learning error, but the interpretation of local models (1) and (2) is inappropriate. On the other extreme, the local models (1) and (2) in Figure 3-10(d) have best interpretability given a high degree of localization ($ld = 10$). Local model (4) in Figure 3-10(d) is misinterpreted to the neighbouring region; however, if the observation is focused on a small regional subspace, it can be argued that the output data values are almost constant within this region. This coincides with the learning result of local model (4) in Figure 3-10(d), which is almost a constant term model. The results of Figure 3-10(b) and (c) are in between the two extreme cases in Figure 3-10(a) and (d), this is made possible with smaller settings of the degree of localization for these models. By visual inspection, if the influence is confined to a regional subspace proportional to the weight (matching degree) of the local models as shown in Figure 3-10(b), then the FITSK¹-LKF model with degree of localization 2 has the best local learning accuracy. The local models for this model have a good balance between learning accuracy and model interpretability.

Figure 3-11: Learning errors of FITSK¹ with different degree of localization.

The learning results of different FITSK¹ settings are illustrated with four different error criteria in Figure 3-11. The criteria are the *Global_MSE* (equation (3-34)), *Local_MSE2* (equation (3-31)), *2CN_MSE* (equation (3-33) with $K=2$), and *Local_MSE1* being defined by [Johansen 2003] as shown in equation (3-37). The *Local_MSE1* is similar to *Local_MSE2* except that the weight of *Local_MSE2* is of order 2.

$$Local\_MSE1 = \frac{1}{P} \sum_{n=1}^N \sum_{p=1}^P w_n^{(p)} \delta_{n,p}^2 \quad (3-37)$$

The learning errors of FITSK¹-GKF are plotted in $ld = 0$, and $ld = [1,10]$ represents the FITSK¹-LKF models with the degree of localization ranging from 1 to 10 accordingly. As shown with the *Global_MSE* criterion, FITSK¹ with global Kalman filter learning has the best global learning accuracy. However, the accuracy gradually reduces as the degree of localization increases. On the other hand, the *Local_MSE1* and *Local_MSE2* criteria demonstrate that the localized Kalman filter learning with degree of localization 2 has the optimal locally weighted least squares error. The *2CN_MSE* criterion is the strictest formulation of the general *KCN_MSE* criterion. It uses the two data points nearest to the centre of a local model to measure the interpretability of a local model. The *2CN_MSE* criterion illustrates that, the interpretability of the local models increases in proportional to the degree of localization. This coincides with Figure 3-10(d), which shows that the local models have best interpretability to the smallest confined region at the centre of the local models.

These conclude that equation (3-33) is a good indicator of interpretability, and equation (3-31) is a good criterion to measure the locally weighted least-squares accuracy of the local models. The benchmarking result demonstrates that the FITSK¹ model with localized Kalman filter can be easily reformulated to have different degree of interpretability and local model accuracy by varying the degree of localization; and is flexible to the system designer based on different design considerations.

3.3 Summary Remarks

This section proposes a generic TSK framework, the FITSK framework. The framework is subsequently mapped to a zero-order FITSK model and a first-order FITSK model. The zero-order and first-order FITSK models assume online parameter learning formulae. In this way, the FITSK models are suitable for adaptive system learning and nonlinear system estimation.

The key strength of the FITSK framework is the flexibility of adopting different formulations and settings to achieve different design criteria. The three design criteria are 1) interpretability; 2) global accuracy; 3) immunity to curse-of-dimensionality. The zero-order FITSK model is suitable for fast adaptive systems without a major consideration of interpretability. The first-order TSK model with global Kalman filter is appropriate for low dimensionality, low interpretability systems with optimal global learning accuracy. A proposed localized version of the Kalman filter algorithm has been shown to achieve optimal accuracy for locally weighted least-squares error criterion. Furthermore, the localized Kalman filter has been demonstrated to have superior immunity to the global Kalman filter on the basis of the curse-of-dimensionality issue. The first-order FITSK model adopting this localized Kalman filter can be readily applied to higher-dimensionality system with lower computational resources when compared against one using global Kalman filter learning. The degree of localization in the localized Kalman filter provides a flexibility to balance the interpretability and system accuracy in the local models. A guideline is discussed based on these three design criteria for model selection of these FITSK models.

The performance of the FITSK models have been successfully evaluated using five nonlinear system benchmarking problems: 1) a nonlinear system; 2) human operation of a chemical plant; 3) daily price of a stock; 4) Mackey-Glass time series prediction; and 5) motorcycle crashing problem. The first three benchmarks demonstrate that the zero-order FITSK model has encouraging performances in terms of learning error and correlation measure. The 4th benchmark highlights the computational advantages of the localized Kalman filter over the first-order FITSK model. The final benchmark illustrates the modeling flexibility of the first-order FITSK model to tackle the balance between local model interpretability and learning accuracy.

A promising extension is to simplify the localized Kalman filter algorithm (equation (3-32)) to be used with the zero-order FITSK parameter learning. Consider simplifying equation (3-24) to be used to tune a zero-order parameter, $b_{0,(l,m)}$, of an arbitrary local model indexed by (l,m) , as shown in equation (3-38).

$$\begin{bmatrix} w_l^{(1)} \\ w_l^{(2)} \\ \vdots \\ w_l^{(p)} \\ \vdots \\ w_l^{(P)} \end{bmatrix} [b_{0,(l,m)}] = \begin{bmatrix} w_l^{(1)} d^{(1)} \\ w_l^{(2)} d^{(2)} \\ \vdots \\ w_l^{(p)} d^{(p)} \\ \vdots \\ w_l^{(P)} d^{(P)} \end{bmatrix} \quad (3-38)$$

Similarly, the derivation for this zero-order parameter tuning algorithm follows equations (3-25) to (3-32). The result is shown in equation (3-39).

$$\begin{aligned} B_{p+1} &= B_p + (w^{(p)})^{ld} S_{p+1} a_{p+1}^T \delta_{p+1}, \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{\frac{1}{(w^{(p)})^{ld}} + a_{p+1} S_p a_{p+1}^T}, \quad p = 0, 1, \dots, P-1 \end{aligned} \quad (3-39)$$

with initial conditions of $B_0 = 0$ and $S_0 = \gamma$, where γ is a large positive number. However, since B_p is a single element array consisting of only the zero-order tunable parameter, S_p matrix similarly follows to be a single element matrix, and a_p is equal to one as easily derived from equation (3-25). Therefore, equation (3-39) is readily simplified into equation (3-40).

$$\begin{aligned} B_{p+1} &= B_p + (w^{(p)})^{ld} S_{p+1} \delta_{p+1}, \\ S_{p+1} &= S_p - \frac{(S_p)^2}{\frac{1}{(w^{(p)})^{ld}} + S_p} \end{aligned} \quad (3-40)$$

This equation can be viewed as a zero-order local learning algorithm with a form of momentum to accelerate the learning process. The momentum is derived by the update equation involving S_p . The parameter B_p is subsequently updated by involving the new momentum term S_{p+1} . The parameter tuning with equation (3-40) is repeated for each local model indexed by (l, m) .

Figure 3-12: Learning results of direct vs. momentum update formulae.

Figure 3-12 shows the learning results (training and testing) of the FITSK⁰ model with the two different update formulae; namely: the direct update (equation (3-14)); and the momentum update (equation (3-40)). The dataset is the motorcycle crash dataset [Hardle 1990] as described in Section 3.1.2. Both of the learning setups have learning rates of 0.2, standard deviation of 0.4247 times of the resolution, and 20 membership functions at layer 2 of FITSK; the localization degree of the momentum update is 2 ($ld = 2$) and $S_0 = 1000000$. The results demonstrate that the momentum update has better and faster learning convergence than the direct update. The momentum update converges within two learning epochs, while the direct update converges after the fifth learning epochs. Similarly, the testing error of the momentum update is optimal after the first learning epoch, in contrast to the fifth learning epochs of the direct update. Future work is to be done on applying this momentum update on other models and other applications. This is to be further compared against other momentum learning formulae for zero-order local models as described in [Ang 2000].

The FITSK framework is naturally extensible to non-singleton input dimension following the formulations described earlier. However, the two models presented in this section both assume singleton input formulation, and thus the structure of the two FITSK models is similar to the ANFIS model [Jang 1992] following layer 2 operations. Although the focus of the FITSK is in its superiority in parameter learning and model interpretability, a possible future work is to consider the natural extension of FITSK to non-singleton system and its corresponding clustering algorithms. This is especially useful for modeling of noisy system. Some insights into this area were highlighted in several literatures [Mouzouris 1997a], [Mouzouris 1997b].

The final possible future work in this section is to automate the model selection of FITSK, which is described in Figure 3-6. This is especially important in automatically determining whether the system is likely to suffer from the curse-of-dimensionality issue. Currently, this process is performed manually based on the subjective perception of the designer. A formal approach should be adopted with supporting studies in the future.

As discussed earlier, the zero-order FITSK model is superior in modeling of fast-adaptive, incremental, online learning system. The next section will incorporate a zero-order FITSK model to assume the function approximation role of the reinforcement learning system. Subsequently, simulations are performed to demonstrate the feasibility of the proposed fuzzy reinforcement learning model.

4 FITSK-RL: Function Approximation with FITSK for a Generic Reinforcement Learning Framework

*"It is not the strongest of the species that survives, nor the most intelligent that survives.
It is the one that is the most adaptable to change."
~ Charles Darwin*

Reinforcement learning is a computational learning approach to optimize the action selection strategy towards some given situational states. Generally, the criteria for maximization are the total future rewards or long-term goals [Kaelbling 1996], [Sutton 1998]. Reinforcement learning model often constitutes by two learning problems, which are the temporal credit assignment problem and the structural credit assignment problem [Sutton 1984]. The temporal credit assignment problem involves in assigning the delayed credit or blame to states or actions in a temporal sequence, and has been well studied in several research efforts. Some popular temporal credit assignment algorithms are the TD(λ) [Sutton 1988], Q-learning [Watkins 1989], and SARSA learning [Rummery 1995], [Sutton 1998]. In contrast, the structural credit assignment tackles the issue of propagating the awarded credit or blame in a particular state or action, to be stored in some internal structures or memories.

However, the usage of simple tabular storage as the internal structure is inadequate due to the Bellman's *curse of dimensionality and modeling* [Bellman 1957]. This has been depicted earlier in Section 1 and 2 of the Thesis. Therefore, a function approximator is critically needed to generalize the return value function during the reinforcement learning process [Bertsekas 1996]. Despite earlier significant failures in practice [Boyan 1995a], [Baird 1995], [Gordon 1995a], some research efforts demonstrated that value function approximations are still possible in practice [Tsitsiklis 1996], [Sutton 1996b], and in theory [Tsitsiklis 1997], [Van Roy 1998]. Van Roy's thesis [Van Roy 1998] demonstrates these successful examples by employing the feature-based methods or the local basis functions for value function approximation. However, the adoption of these local basis functions is unsystematic, and often involves a trial-and-error process on the selections of local functions [Van Roy 1998].

On the other hand, several fuzzy inference systems have been employed as value function approximators [Berenji 1992], [Glorennec 1994], [Berenji 1996], [Lin 1996a], [Jouffe 1998], [Kim 1999]. These systems have been demonstrated to be effective in the generalization of value functions in continuous state-space reinforcement learning problems. Furthermore, the

fuzzy inference schemes adopted in these systems are systematically defined, and have strong theoretical foundation from the fuzzy logic perspective. However, the roles of fuzzy inference systems in these models are weakly described and are not modularly formulated. Furthermore, most of these models assume a particular temporal learning algorithm, and are not applicable to general classes of reinforcement learning algorithms. Please refer to [Quah 2004a], or Section 2.2.3 of this Thesis on a survey of these fuzzy reinforcement learning models.

Hence, a two-stage approach to formulate a modular fuzzy reinforcement learning system that is immune to the above-mentioned deficiencies is proposed in this section. The proposed fuzzy reinforcement learning system is termed the *fuzzy input Takagi-Sugeno-Kang for generic reinforcement learning* (FITSK-RL). The first stage involves formulating a generic reinforcement learning framework with strong modularity and functional abstraction. The function approximator, policy generator and TD generator are three modules constituting the generic framework. This provides a functional independency among the three modules. Furthermore, the framework can assume several popular temporal learning algorithms, such as the TD(λ), Q-learning, and SARSA learning, with some minor alterations by switching to different action-selection policies. Therefore, any possible function approximator can suitably be employed in this framework to cater for the three general classes of temporal learning algorithms. The second stage of this modular design involves formulating a zero-order fuzzy input Takagi-Sugeno-Kang (FITSK) model [Quah 2004b], proposed earlier in Section 3 of this Thesis, as the function approximator module in this framework. The zero-order FITSK model is a special case of the local basis function approximation. Thus the theoretical results from Tsitsiklis and Van Roy [Tsitsiklis 1997], [Van Roy 1998] for local basis function approximation are readily extensible to the FITSK-RL model. However, the FITSK-RL model is immune to the deficiency present in the ad-hoc formulation of the local basis function approximation method. This is achieved by having a set of well-defined and systematic operations from the fuzzy inference scheme perspective. Finally, a version of FITSK-RL model is proposed to resolve the continuous action reinforcement learning problem. This ensures that the FITSK-RL model is generic and is applicable to discrete state-action, discrete state-continuous action, and continuous state-action reinforcement learning problems.

This section is organized as follows. Section 4.1 describes a generic reinforcement learning framework; its modular design and general functions. Section 4.2 presents the FITSK-RL

model that has been developed by mapping a zero-order FITSK model onto the function approximator module of the generic framework. In Section 4.3, the FITSK-RL model is evaluated using two different benchmarking simulations. Section 4.4 summarizes this section.

4.1 A Generic Reinforcement Learning Framework

Figure 4-1: A generic reinforcement learning framework.

A generic reinforcement learning framework consists of a generic agent interacting with the external environment. The generic agent adopts a modular design consisting of three modules; namely: function approximator, policy generator, and temporal difference (TD) generator, as shown in Figure 4-1. The function approximator module predicts the value functions based on the state information. The policy generator module generates an appropriate action and the corresponding action-value. The TD generator module subsequently formulates the temporal difference error based on the reinforcement signal, current action-value, and successive action-value. Consequently, the temporal difference error is feedback to the function approximator module to adjust the tunable parameters.

Two motivations drive the modular design of the generic reinforcement learning framework; they are namely:

- to create a generalized framework whereby different reinforcement learning models such as TD learning, SARSA learning, or Q-learning can be mapped with ease onto such a framework; and
- to provide functional independencies and abstractions to each module such that reformulation of a single module would not affect the operations of the other modules.

For example, it is possible to implement the function approximator as a neural network, basis function, or other possibilities without changing the design of the policy generator.

4.1.1 Functionality of Modules

Function approximator:

The function approximator is a multiple-input-multiple-output (MIMO) system to predict the action-value vector with a given state vector and a tunable parameter vector. It implements the function mapping of, $f_{FA} : S \times \mathbb{R}^K \mapsto \mathbb{R}^{|A(s)|}$, as described by equation (4-1).

$$Q_t(s_t, a) = f_{FA}(s_t, c'), \forall a \in A(s_t) \quad (4-1)$$

where

- t is the time index,
- K is the number of tunable parameters of the function approximator,
- $A(s_t)$ is the set of all possible actions at state s_t ,
- $f_{FA}(\cdot)$ is the function approximator's output,
- $c' \in \mathbb{R}^K$ is the tunable parameter vector at time t , and
- $Q_t(\cdot)$ is the action-value vector with cardinality $|A(s_t)|$ at time t .

Similarly, the action-value vector at the next state is given by equation (4-2).

$$Q_t(s_{t+1}, a) = f_{FA}(s_{t+1}, c'), \forall a \in A(s_{t+1}) \quad (4-2)$$

Consequently, the tunable parameters are updated with respect to the state information and the temporal difference error given in equation (4-3).

$$\Delta c' = f_c(s_t, \delta_t) \quad (4-3)$$

where $f_c(\cdot)$ implements the error function of the tunable parameter.

Policy generator:

The policy generator module selects the appropriate action and its corresponding action-value from the given action-value vector based on some predefined policies. The policy generator consists of two components; namely: the *behavior policy*, and the *estimation policy*. The behavior policy generates the actual action; the estimation policy evaluates and improves the current estimation [Sutton 1998]. The generic formulation of the behavior policy is a function of the given action-value vector, as described by equation (4-4).

$$a_t = f_{BP}(Q(s_t, a)) \quad (4-4)$$

where $f_{BP}(\cdot)$ is the action-selection strategy of the behavior policy. The action-value, $Q(s_t, a_t)$, is subsequently retrieved based on the selected-action, a_t .

Similarly, the estimation policy implements a function of the subsequent state action-value vector described by equation (4-5).

$$a' = f_{EP}(Q(s_{t+1}, a)) \quad (4-5)$$

where $f_{EP}(\cdot)$ is the action-selection strategy of the estimation policy. The action-value, $Q(s_{t+1}, a')$, is similarly retrieved on the basis of the selected-action, a' .

Some possible action-selection strategies are listed in equation (4-6).

$$\begin{aligned} \text{Random} : f_{\text{random}}(Q(s_t, a)) &= \underset{\forall a \in A(s_t)}{\text{rand}}(a) \\ \text{Greedy} : f_{\text{greedy}}(Q(s_t, a)) &= \arg \max_{\forall a \in A(s_t)} Q(s_t, a) \\ \epsilon\text{-greedy} : f_{\epsilon\text{-greedy}}(Q(s_t, a)) &= T(p_t > \epsilon) \times f_{\text{greedy}}(Q(s_t, a)) + T(p_t \leq \epsilon) \times f_{\text{random}}(Q(s_t, a)) \\ \text{Softmax} : f_{\text{softmax}}(Q(s_t, a)) &= \arg \max_{\forall a \in A(s_t)} \frac{e^{Q(s_t, a)}}{\sum_b e^{Q(s_t, b)}} \end{aligned} \quad (4-6)$$

where

- $\text{rand}(\cdot)$ is the random selection function,
- ϵ is a user-specify probability measure,
- p_t is a randomly generated probability measure at time t , and
- $T(\cdot)$ outputs a numerical value one for true condition, and zero otherwise,

Temporal difference generator:

The temporal difference (TD) generator computes the temporal difference error based on the immediate reward, current action-value, and successive action-value estimation and is described in equation (4-7).

$$\delta_t = f_{TD}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) \quad (4-7)$$

where $f_{TD}(\cdot)$ is a generic function of the TD generator module.

The generic TD function is extended to the TD error of the cumulative discounted reward formulation using equation (4-8).

$$\delta_t = f_{TD}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) = r_{t+1} + \gamma Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \quad (4-8)$$

where γ is the discount rate parameter defined in Section 2.1.1.

4.1.2 Mapping to Autonomous and Controlled Models

Section 4.1.1 described the general functions of a reinforcement learning framework. However, the mapping of various popular reinforcement learning models into this framework has yet to be formulated. This subsection attempts to show how the generic reinforcement learning framework can be readily reformulated as TD learning, SARSA learning, and Q-learning.

TD learning:

An autonomous system is a passive system where the state transitions are not affected by the decisions or actions taken [Van Roy 1998]. The action selection process is irrelevant in an autonomous system. Consequently, all the actions can be excluded from consideration in the generic reinforcement learning framework for an autonomous system. The output of the function approximator reduces to the prediction of a single state-value function instead of the action-value vector, as described by equation (4-9).

$$\begin{aligned} V_t(s_t) &= f_{FA}(s_t, c') \\ V_t(s_{t+1}) &= f_{FA}(s_{t+1}, c') \end{aligned} \quad (4-9)$$

Similarly, the policy generator is irrelevant in an autonomous system. The state-values from equation (4-9) are directly forwarded to the TD generator. Equation (4-8) is subsequently reduced to the TD error of temporal difference learning given by equation (4-10).

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \quad (4-10)$$

Contrary to an autonomous system, the controlled system is an active system whereby the system's dynamics are directly influenced by the control decisions and policies [Van Roy 1998]. There are two classes of controlled reinforcement learning models, namely: the on-policy and the off-policy TD models. The on-policy model uses an identical policy to generate control behavior and to estimate the optimal policy. On the contrary, the off-policy model assumes different behavior and estimation policies [Sutton 1998].

SARSA learning:

A popular on-policy TD control model is the SARSA model [Rummery 1994]. The behavior policy is identical to the estimation policy in this model, as shown in equation (4-11).

$$\begin{aligned} a_t &= f_{EP}(Q(s_t, a)) = f_{BP}(Q(s_t, a)) \\ a' &= a_{t+1} = f_{EP}(Q(s_{t+1}, a)) = f_{BP}(Q(s_{t+1}, a)) \end{aligned} \quad (4-11)$$

and

$$Q(s_{t+1}, a') = Q(s_{t+1}, a_{t+1}) \quad (4-12)$$

Given equations (4-8) and (4-12), the TD error of SARSA learning is computed by equation (4-13).

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \quad (4-13)$$

Q-learning:

Q-learning is an off-policy TD control model proposed by [Watkins 1989]. The estimation policy of Q-learning is a greedy policy, which is different from the behavior policy and is described by equation (4-14).

$$\begin{aligned} a' &= f_{EP}(Q(s_{t+1}, a)) = f_{\text{greedy}}(Q(s_{t+1}, a)) = \arg \max_{\forall a \in A(s_{t+1})} Q(s_{t+1}, a) \\ Q(s_{t+1}, a') &= Q(s_{t+1}, \arg \max_{\forall a \in A(s_{t+1})} Q(s_{t+1}, a)) = \max_{\forall a \in A(s_{t+1})} Q(s_{t+1}, a) \end{aligned} \quad (4-14)$$

Given equations (4-8) and (4-14), the TD error of Q-learning is computed accordingly as shown in equation (4-15).

$$\delta_t = r_{t+1} + \gamma \max_{\forall a \in A(s_{t+1})} Q(s_{t+1}, a) - Q(s_t, a_t) \quad (4-15)$$

The above discussion demonstrates the natural extensions of the generic reinforcement learning framework to the three popular TD learning models. With this discussion, it is possible to segregate the formulation of the function approximator and the TD generator modules from the policy generator module. Further reformulation of the function approximation module is readily extensible to these three popular TD learning models.

4.2 FITSK-RL Model

The Fuzzy Input Takagi-Sugeno-Kang fuzzy framework (FITSK) proposed earlier in Section 3 and in [Quah 2004b] is a framework that assumes some local model representations with local function dynamics at the consequent or rule-layer of the framework. This framework computes the output by performing fuzzy interpolations of these simpler local functional models in the neighboring fuzzy partitions. The general motivations for this framework are to

achieve a high learning accuracy with online learning algorithm, and to maintain a good interpretability of the local models for nonlinear system estimations.

A zero-order FITSK model is a special case of the FITSK framework by assuming the local model functional unit being of zero-order dynamics. The model adopts a systematic modeling approach based on the Takagi-Sugeno-Kang fuzzy system [Sugeno 1988], [Takagi 1985]. This section integrates the zero-order FITSK model into the generic reinforcement learning framework proposed in the previous section. The zero-order FITSK model assumes the role of the function approximator module to estimate the value functions based on the given state information. The simplicity of its parameter learning formulae positions it as an excellent candidate for learning of fast adaptive system, such as in the reinforcement learning system. The resultant model is termed as the *Fuzzy Input Takagi-Sugeno-Kang for Generic Reinforcement Learning* (denoted by FITSK-RL).

This subsection is organized as follows. Section 4.2.1 briefly summarizes the zero-order FITSK model from the literature. Section 4.2.2 integrates the zero-order FITSK model as the function approximator for the generic reinforcement learning framework for a discrete action reinforcement learning problem context. Section 4.2.3 extends the FITSK-RL model to a continuous action reinforcement learning problem domain.

4.2.1 The Zero-order FITSK model

The zero-order FITSK model [Quah 2004b] has the same structure as the FITSK framework with zero-order local functional unit (constant term). The FITSK model is a six-layer fuzzy system assuming Takagi-Sugeno-Kang (TSK) fuzzy inference scheme [Sugeno 1988], [Takagi 1985]. The TSK inference scheme provides a strong theoretical foundation to the FITSK model from the fuzzy modeling perspective.

Given an input state vector, $s \in \mathfrak{R}^N$, with N being the dimension of the state vector, and a weight vector, $w \in \mathfrak{R}^K$, with K being the total number of local models. A membership function, $\mu(s) : \mathfrak{R}^N \mapsto \mathfrak{R}^K$, is a projection of the state vector onto the K -dimensional space. A weight is defined as the normalized value of the membership function's output. Summarizing these with the operations from equations (3-1) to (3-7) from Section 3.1.1, the weight vector is computed with equation (4-16).

$$w_k = \frac{\mu_k(s)}{\sum_{j=1}^K \mu_j(s)} \quad (4-16)$$

where

$s = (s_1, \dots, s_N)^T$ is the input state vector,

k is the index of the weight,

$w = (w_1, \dots, w_K)^T$ is the weight vector, and

$\mu(s) = (\mu_1(s), \dots, \mu_K(s))^T$ is the vector of membership function.

For an L -output zero-order FITSK system, the computed output is a vector $y \in \Re^L$, indexed by $l = (1, \dots, L)$; associated with each output is a vector of tunable parameters, $c_l \in \Re^K$, and the output y_l is computed as described in equation (4-17).

$$y_l = w^T c_l = \sum_{i=1}^K w_i c_{i,l} \quad (4-17)$$

where $c_l = (c_{1,l}, \dots, c_{K,l})^T$, and $c = (c_1, \dots, c_L)$, $c \in \Re^{K \times L}$ is the matrix of the zero-order functions (constant terms), see also equation (3-13) for details.

The direct update formula for each tunable parameter is computed as equation (4-18).

$$c_{k,l}^{t+1} = c_{k,l}^t + \alpha w_k' (d_l^t - y_l^t) \quad (4-18)$$

where

t denotes the time index,

α is the learning constant, and

d is the desired output vector, $d = (d_1, \dots, d_l, \dots, d_L)$.

A technique to manage the standard deviation of the membership function is employed to control the bias-variance of the system, which is defined in equation (4-19).

$$\sigma \propto sr^{ld} \quad (4-19)$$

where

$sr \in \Re$ is the shrinking rate and $sr \in (0, 1]$, and

ld is the degree of localization.

A simple experiment was conducted in Section 3.1.2 to illustrate that the degree of localization is directly proportional to the bias of the system. Therefore, a higher degree of localization refers to a higher accuracy (bias) and lesser generalization (variance), and vice-

versa. A simple formulation can be devised by equating the degree of localization to the current learning episode of a reinforcement learning system, such that the learning commences with better generalization and gradually moves towards higher learning accuracy. This is illustrated with a simulation in Section 4.3.2, whereby the system with this formulation has the advantage of advancing towards a better policy.

This subsection summarizes the general operations of the zero-order FITSK model. Please refer to Section 3.1.2 for further technical details and formulations of the model.

4.2.2 Discrete Action FITSK-RL

Previous subsection summarizes the operations of the zero-order FITSK model. This subsection will extend the model to assume the function approximator role of the generic reinforcement learning framework proposed in Section 4.1.

The structure of the model is an N -input, L -output system described in the previous subsection, with N being the number of state dimension, and L being the number of discrete actions. Each output of the model predicts an action-value associated with the state-action pair (s, a) . Therefore, for an input state vector, s , and a vector of all possible discrete actions, $a = (a_1, \dots, a_L)$; the output of the model is a vector, $Q(s, a) \in \mathbb{R}^{|A(s)|}$, with one action-value for each discrete action. The output vector of the network is computed directly with equation (4-17). The learning process includes the following two formulations; namely:

- the derivation of the eligibility trace update formulae, and
- the integration of TD learning into the function approximator.

Eligibility traces:

The eligibility trace was proposed by [Sutton 1984] to accelerate the temporal learning process of delayed reinforcement learning systems. The eligibility trace is a vital component of the popular TD(λ) reinforcement learning algorithm [Sutton 1988]. The eligibility trace formula assigns time-decaying traces to recently visited states, and the error is propagated to states with non-zero traces. This formulation accelerates the temporal learning process as delayed rewards (to be discovered in the near future) can directly affect the values of past states.

The formulation of eligibility trace for the FITSK model is similar to the basis function approximation [Van Roy 1998]. Defining $e \in \mathbb{R}^{K \times L}$ as the eligibility trace matrix associated

with the local models of FITSK (i.e., each node in layer 5 of the FITSK model holds an eligibility trace vector). The *accumulating eligibility trace* update formula at time t is shown in equation (4-20) (see also Section 2.1.2 for basic eligibility trace formulations).

$$e_t = \sum_{\tau=0}^t (\gamma\lambda)^{t-\tau} w(s_t, a_t) \quad (4-20)$$

where

- e_t is the eligibility trace vector at time t ,
- λ is the trace-decay parameter, *gamma*, with value range $[0,1]$, and
- w is the weight vector associated with (s_t, a_t) , as defined in equation (4-16).

The recursive update formula for equation (4-20) is represented in equation (4-21).

$$e_t = \gamma\lambda e_{t-1} + w(s_t, a_t) \quad (4-21)$$

and the initial condition is given as $e_0 = w(s_0, a_0)$ [Van Roy 1998].

The TD(λ) learning adopting the accumulating eligibility trace formula (equation (4-20) or (4-21)) has a biased estimation of the learning value. In contrast, the *replacing eligibility trace* update formula results in a TD(λ) learning with unbiased value function estimation [Singh 1996]. The replacing eligibility trace update formula at time t is defined in equation (4-22) and its recursive version in equation (4-23).

$$e_t = \max_{\tau=0}^t ((\gamma\lambda)^{t-\tau} w(s_t, a_t)) \quad (4-22)$$

$$e_t = \max(\gamma\lambda e_{t-1}, w(s_t, a_t)) \quad (4-23)$$

and the initial condition is given as $e_0 = w(s_0, a_0)$.

Integrated TD learning:

The parameter tuning of the FITSK-RL model is achieved by incorporating the eligibility trace formula and TD error into the direct update formula of equation (4-18) and is described in equation (4-24).

$$c^{t+1} = c^t + \alpha e_t \delta_t \quad (4-24)$$

where

- c is the tunable parameter matrix described in equation (4-17), and
- δ is the numerical-valued TD error.

If the reinforcement learning assumes a form of the autonomous TD learning system, as described in Section 4.1.2, then the action selection is irrelevant and the number of FITSK output is given as $L = 1$. Therefore, both the eligibility trace and the tunable parameter matrices reduces to K -dimensional vectors. The TD error is computed by equation (4-10). The accumulating and replacing eligibility trace update formulae are simplified into equations (4-25) and (4-26) respectively.

$$e_t = \gamma e_{t-1} + w(s_t) \quad (4-25)$$

$$e_t = \max(\gamma e_{t-1}, w(s_t)) \quad (4-26)$$

In contrast, the TD error is computed using equation (4-13) or (4-15) for controlled TD learning systems. This is dependant on whether it is a SARSA or a Q-learning model.

There are two motivations for having the FITSK model as the function approximator for the reinforcement learning system. Firstly, the reinforcement learning problem domain is extended to a continuous state-space. Secondly, the FITSK model provides a means to systematically craft the function approximation process with the well-defined TSK fuzzy inference scheme. Fuzzy inference systems have been employed as value function approximators in the following papers [Berenji 1992], [Glorennec 1994], [Berenji 1996], [Lin 1996a], [Jouffe 1998], [Kim 1999]. These systems have been demonstrated to be effective in the generalization of value functions in continuous state-space reinforcement learning problems. However, the roles of fuzzy inference systems in these models are weakly described.

On the other hand, Van Roy has proven the convergence and stability issues for reinforcement learning systems based on a linear *local basis function* approximation method [Van Roy 1998]. From Section 4.2.1, it is easy to deduce that the zero-order TSK fuzzy system is a special case of the local basis function approximation. Thus, the theoretical convergence and stability results are directly extendible from Tsitsiklis and Van Roy's results [Tsitsiklis 1997], [Van Roy 1998], given that certain assumptions are met. However, the local basis function method lacks a systematic and broadly defined guideline for the selection of the basis functions. This is accommodated by the precise and systematic TSK inference scheme present in the FITSK model, as described in Section 3.1 and summarized in Section 4.2.1.

The theoretical convergence results of reinforcement learning for linear basis function approximation methods [Van Roy 1998] are summarized in the following statements.

- The $TD(\lambda)$ of autonomous system has guaranteed convergence property. Therefore, AHC or actor-critic methods [Sutton 1984], [Sutton 1998] similarly possess guaranteed convergence property.
- The controlled $TD(\lambda)$ system with on-policy learning (i.e. SARSA learning) are generally converging provided that the policy will eventually settle into a limiting policy.
- The controlled $TD(\lambda)$ system with off-policy learning (i.e. Q-learning) has no guarantee of convergence.
- All reinforcement learning methods with nonlinear function approximation (i.e. backpropagation neural-network and higher-order basis function) are diverging.

Given that the zero-order TSK system, including the zero-order FITSK model, is a special case of the local basis function method, the theoretical convergence results are directly applicable to these learning systems. This implied that fuzzy reinforcement learning systems with actor-critic methods or SARSA learning have good convergence results. However, fuzzy Q-learning [Berenji 1996], [Glorennec 1997], [Kim 1999] has no guarantee of convergence. Value function approximations with higher-order TSK systems are generally diverging.

4.2.3 Continuous Action FITSK-RL

Previous subsection described the formulation of the FITSK model for function approximation of discrete action reinforcement learning. However, there is some reinforcement learning tasks that have inherent continuous action variables. The discrete action FITSK-RL model is not applicable to these tasks. This is resolved by extending the discrete action FITSK-RL to the continuous action domain. The formulation of continuous action FITSK-RL model involves three aspects; namely:

- the formulation of the action-value as a function of a single continuous action,
- the optimization of action-value for a single continuous action, and
- the extension to multiple continuous actions.

Single continuous action FITSK-RL structure:

The structure of the function approximator for a single continuous action FITSK-RL model is an $(N+1)$ -input, single-output system; with N being the number of state dimensions, the other dimension stores the single continuous action. The single output of the model predicts a function of the continuous action-value associated with the state-action pair (s,a) . Therefore,

given a particular input state vector, s , the output of the model is $Q(s, a)$. $Q(s, a)$ is a function of the single continuous action variable, $a \in \mathcal{R}$. Given a state vector and a parameter vector, the function approximator implements a function of an action at time t as described in equation (4-27).

$$Q_t(s_t, a) = f_{FA}(s_t, c^t), a \in \mathcal{R} \quad (4-27)$$

Action-value optimization method:

The function in equation (4-27) is employed with an optimization method to search for the best action based on some policies. The optimization method searches through the range of the single continuous action variable, and chooses the best action based on the computed action-value. A greedy policy is implemented by searching through the range of the action-value, and choosing the optimal action that is associated with the maximum action-value, as described in equation (4-28).

$$f_{\text{greedy}}(Q(s_t, a)) = \arg \max_{a \in \mathcal{R}} Q(s_t, a) \quad (4-28)$$

The random and ϵ -greedy policies are formulated accordingly to equation (4-6).

The maximum of the function is determined by a single variable root-finding algorithm proposed by Brent [Brent 2002]. This root-finding algorithm is a single variable optimization method without the need of explicitly computing the derivative of the variable. The algorithm is a combination of golden section search and successive parabolic interpolation methods with an optimal bracketing interval governed by the value of 0.381966 (or $(3 - \sqrt{5})/2$, which is the golden section). Please refer to Chapter 5 of [Brent 2002] for details of the algorithm.

Multiple continuous actions FITSK-RL:

Consider a multiple continuous actions reinforcement learning task, with N being the state dimensions and L being the number of continuous action variables. A direct presumption is to extend the single continuous action FITSK-RL model into an $(N+L)$ -input, single-output system. Subsequently, the Q -function is represented as a multi-variable function of the action variables, and the function maximization is computed with a multi-variables optimization algorithm. However, this presumption is not realizable due to the following two reasons; namely: a) the severe curse-of-dimensionality of the high-dimensional $(N+L)$ -input system, and b) the multi-variable function has a larger searching space, the searching process is easily trapped in local maxima, resulting in inaccurate global optimization. Therefore, this

subsection will discuss a simpler and realizable design for a multiple continuous actions reinforcement learning task.

Denoting the multiple continuous actions as a vector, $a = (a_1, \dots, a_L)$, indexed by $l = \{1, \dots, L\}$. The function approximation in FITSK-RL is represented by L parallel function approximators. Each function approximator assumes the structure of a single continuous action FITSK-RL approximator as described previously. The result of this design is a parallel structure of L $(N+1)$ -input, single-output systems, one for each continuous action variable. The operations of these parallel systems are identical to the single continuous action FITSK-RL model. Then, an arbitrary continuous action, a_l , is similarly selected with equation (4-29) for a greedy policy.

$$f_{\text{greedy}}(Q(s_t, a_l)) = \arg \max_{a_l \in \mathcal{A}} Q(s_t, a_l) \quad (4-29)$$

The best action among these multiple continuous actions, a^* , is chosen by a linear search through these maximum action-values.

$$a^* = \arg \max_{\forall l \in \{1, \dots, L\}} f_{\text{greedy}}(Q(s_t, a_l)) \quad (4-30)$$

Brent's optimization method [Brent 2002] was first adopted in the HEDGER algorithm [Smart 2000] to search for the best continuous action and its associated action-value. HEDGER algorithm is a TD(0) learning model, and using *locally weighted regression* (LWR) to implement the function approximation of the state-action values. This algorithm is demonstrated to be effective in continuous state, single continuous action tasks. There are two differences between the continuous action FITSK-RL model and the HEDGER algorithm. They are: a) The FITSK-RL model adopts a generic reinforcement learning framework with general TD(λ) algorithm. However, the HEDGER algorithm is focused on TD(0) learning with the Q-learning algorithm. b) The HEDGER algorithm has to be supplied with an initial knowledge prior to the actual learning trajectories. This is due to the adoption of Lin's experience-replay method [Lin 1992], where experiences are presented to the learner in reverse order after ending a trajectory. The initial learning will suffer if the initial knowledge is erroneous or absent. However, the FITSK-RL model is an online model that learns without the need of an initial knowledge. The function approximator is updated and the policy is corrected at every time step, thus it is immune to the above deficiency.

As a final remark, the FITSK model is introduced in this section to implement the function approximator of the generic reinforcement learning framework described in Section 4.1. Therefore, following the discussion from Section 4.1, the framework is readily extendible to three popular reinforcement learning models. Thus, the FITSK-RL model can be incorporated into TD-learning, SARSA learning or Q-learning systems with a zero-order FITSK model as the function approximator. This result generalizes the FITSK-RL model as a generic fuzzy reinforcement learning model, which is applicable to continuous state, continuous action reinforcement learning domains.

4.3 Simulation Results and Analysis

The FITSK-RL model is evaluated using two reinforcement learning problems: a) the puddle world, and b) the mountain-car task, both found in [Boyan 1995a], [Sutton 1996b]. The detailed dynamics and objectives of these benchmarking simulations are given in the respective subsections. Unless otherwise specified, the parameters listed in Table 4-1 are used in these simulations with on-policy SARSA learning. Please refer to Sections 4.1 and 4.2 for the semantics of the parameters.

Table 4-1: Predefined FITSK-RL parameters

Learning constant, $\alpha = 0.2$	Eligibility trace formula = replacing trace
Shrinking rate, $sr = 0.99$	Trace-decay parameter, $\lambda = 0.5$
Strategy = ϵ -greedy ($\epsilon = 0.01$)	Discount rate parameter, $\gamma = 0.999$

Figure 4-2: Aspects of FITSK-RL for benchmarking simulations

As depicted in Figure 4-2, the FITSK-RL model with SARSA learning is simulated with a simple discrete state and discrete action task; namely: the puddle world task. The puddle world task is modified from the original formulation [Boyan 1995a], and is described in Section 4.3.1. Subsequently, the continuous state versions of the FITSK-RL model are simulated with the mountain car task. There are two sets of dynamics for the mountain car task; namely: the discrete action mountain car, and the continuous action mountain car. Each of these problems is accommodated with different formulations of the continuous state FITSK-RL models respectively; they are the discrete action and the continuous action FITSK-RL models. In the simulation of discrete action continuous action FITSK-RL model, both SARSA and Q-learning are studied and the results subsequently contrasted. These are demonstrated in Section 4.3.2 of the Thesis. The adaptability of the FITSK-RL framework to support different controlled TD models is highlighted from these studies.

4.3.1 Puddle World task

The puddle world [Boyan 1995a], [Sutton 1996b] is a two-dimensional space delayed reinforcement learning task, with each state-space in the range of $[0,1]$. Four discrete actions are available at each time step, namely: up, down, left, and right. Each selected action will then advance towards that particular direction with a distance of 0.05. There are two puddles within the space, located at center points of $[(0.1,0.75), (0.45,0.75)]$, and $[(0.45,0.4), (0.45,0.8)]$, each with a radius of 0.1 as shown by the shaded region in Figure 4-4. If the centre of the puddles were entered, the penalties of -400 were applied; for those with a step away (0.05) from the centre of puddles, the penalties were -200 ; and for those two steps away (0.1), the penalties were reduced to -100 . The rewards for other non-puddle states are: -1 for each non-goal state, and zero for the goal state. The goal state is located at $(1,1)$, and the initial state is located at $(0.25, 0.55)$ in the state space. A 2-input, single-output FITSK-RL model with 5×5 input linguistic nodes is employed to learn the task. Both the behavior and estimation policies are set as ϵ -greedy, resulting in a SARSA learning model.

Figure 4-3: Number of steps to reach goal for each trial.

Figure 4-3 plots the number of steps to reach the stated goal during the 500 trials of learning, with the result being averaged over 10 independent runs. As shown in Figure 4-3, all the trajectories are able to stably converge to near-optimal policies of about 40 steps (optimal policy = 36 steps) after about 200 trials of learning.

Figure 4-4: Trajectories of optimal (36 steps) and sub-optimal (46 steps) policies.

Figure 4-4 plots the actual trajectories of two learning trials from the same initial state, one with an optimal policy and the other with a sub-optimal policy. The grey areas are the locations of the puddles. The optimal policy follows the edge of the puddles and reaches the goal in a minimal number of steps.

Figure 4-5: (a) Cost-to-goal, (b) action dynamics; during the 500th trial of learning.

Figure 4-5 plots the dynamics learnt during the 500th trial of learning starting with the same initial state. Part (a) shows the *cost-to-goal* function ($-\max_a Q(s,a)$) of the corresponding state-space, which is a continuous function with the maximum value located around the puddles, and the minimum value near-to the goal state. Regions with high cost-to-goal values represent the area to be avoided by the agent. Generally, the agent is moving from higher cost-to-goal regions towards lower cost-to-goal regions; and eventually reaching the goal with the lowest cost-to-goal (zero cost-to-goal). This is outlined with a possible trajectory based on the learnt value in Figure 4-5(a). Part (b) plots the corresponding action dynamics of the optimal policy. The four actions are: left (white), right (light-grey), down (dark-grey), and up (black). Some of the actions are inaccurate (left action) because those states are seldom encountered and are lack of training. This is due to the assumption of a fixed initial starting point for all trials. This problem can be easily resolved, by assuming random initial points. At those regions neighboring to the optimal trajectory, the action dynamics are accurate since sufficient learning is acquired in these areas.

This demonstrates the effectiveness of the FITSK-RL model in solving a discrete state, discrete action reinforcement learning task. Comparing with the results in [Sutton 1996b], the CMAC model consisted of 5 tilings, each with 5x5 receptive fields. Therefore, the total memory elements for value storage are 125 nodes. However, the FITSK-RL model has a total of 5x5 local models for the generalization representation of learning results in a continuous manner. This is one-fifth the memory requirement of the CMAC model, or 5-fold

improvements in terms of storage complexity. This demonstrates the effectiveness of employing the fuzzy TSK models for function approximation in reinforcement learning system.

4.3.2 Mountain-Car task

This simulation is conducted to evaluate the effectiveness of FITSK-RL in modelling continuous state and continuous action reinforcement learning problem. Furthermore, SARSA learning and Q-learning are integrated with the generic reinforcement learning framework to be used with the FITSK-RL model. This is used to illustrate that the generic framework can incorporate both the controlled TD systems with ease. The first part of this subsection studies the FITSK-RL model with SARSA and Q-learning, for application of the continuous state, discrete action mountain-car task with formulations from Section 4.2.2. The second part of this subsection experiments the FITSK-RL model for the continuous state, continuous action application domain, with formulations from Section 4.2.3.

Continuous state, discrete action FITSK-RL model:

The FITSK-RL model is simulated with the mountain-car task [Boyan 1995a], [Sutton 1996b]. The task is to drive an underpowered car up a steep mountain road, as shown in Figure 4-6. The difficulty is that the gravity is greater than the car's acceleration achievable by the engine, and even at full throttle, the car cannot directly accelerate up the steep slope. The only solution is to first move away from the goal and up the opposite slope on the left. By combining the inertia built up, the car manages to climb up the slope. The task is chosen due to its significance as a delayed reinforcement learning problem. Training consisted of a series of trials, each starting from a selected non-goal state and continuing until the goal region was reached. On each non-goal step a penalty (negative reinforcement) of -1 was incurred. There are three possible actions: full throttle reverse (-1), zero throttle (0), and full throttle forward (+1). The goal is encountered when the position is greater than 0.5.

Figure 4-6: Mountain Car task.

The dynamics of the task are listed below:

Given a state $s_t = (x_t, v_t)$, where the position is x_t , the velocity is v_t , and an action a_t , at time step t , the next step dynamics are specified in equation (4-31).

$$\begin{aligned} a_t &\in \{-1, 0, 1\}, v_t \in [-0.07, 0.07], x_t \in [-1.2, 0.6] \\ v_{t+1} &= v_t + 0.001a_t - 0.0025 \cos(3x_t) \\ x_{t+1} &= x_t + v_{t+1} \end{aligned} \quad (4-31)$$

A 2-input, 3-output, 15x15 input linguistic node, discrete action FITSK-RL model is employed to learn the task. The experiments begin with setting both the behavior and estimation policies as ε -greedy, resulting in a SARSA learning model.

Figure 4-7: Average number of steps to reach goal for FITSK-RL.

Figure 4-7 shows the simulation results averaged over 10 independent runs starting from the same initial state, each run consists of a total of 2000 trials, the figure shows the average number of steps to reach its goal. As shown in the figure, the FITSK-RL model is converging to a near-optimal policy of about 107 steps after about 1650 trials.

Figure 4-8: Average number of steps to reach goal for tile-coding.

A tile-coding model [Sutton 1996b] is simulated with resolutions of $1/15$ for each dimension in Figure 4-8. This setting is similar to the resolution of the input linguistic nodes of the FITSK-RL model. The learning constant, trace-decay parameter, strategy, and discount-rate parameter are identically set with respect to Table 4-1, and is similarly simulated for 10 independent runs. However, the converged policy of the single-tile, tile-coding model takes an average of 167 steps to reach the goal. This shows a significant policy improvement of about 60 steps in the FITSK-RL model over the tile-coding model despite having similar memory resources. Generally, the policies are improved when the number of tiles is increased due to a higher sampling resolution with better generalization capability. For a setting of 7 tiles, the converged policy of the tile-coding model has similar performance to the FITSK-RL model, which requires about 113 steps to reach the goal. However, the required resources are increased by 7-fold. Therefore, the FITSK-RL model provides a 7-fold improvement in memory resources in comparison with the tile-coding model in modelling of the continuous state mountain-car task. This superior memory requirement is due to the reason that the FITSK-RL model fuzzifies the input dimensions. This relaxes the needs for having finer resolutions in discretizing the input space in order to achieve better accuracy. On the other hand, the fuzzified input spaces span across a wider region, and thus improved the generalization of FITSK-RL to wider neighbouring spaces.

Figure 4-9: Trajectories of near-optimal and sub-optimal policies.

Figure 4-9 plots the trajectories of two policies learnt during a trial starting from the same initial state at $(-0.5, 0)$. The sub-optimal policy has taken 154 steps to achieve the goal. It has not collected sufficient momentum to go up the slope during the first sequence of reverse throttle attempts, and has managed to achieve the goal only by a second attempt. On the other hand, the other near-optimal trial has taken 108 steps to achieve the goal. During the first sequence of reverse driving, the car has collected sufficient momentum to reach the goal.

Figure 4-10: Number of steps to reach goal (without shrinking of membership).

As discussed in Section 4.2.1, the standard deviation of the membership function directly controls the generalization of the system. It was pointed out that by gradually decreasing the standard deviation using a shrinking rate (sr), the system is able to achieve better generalization and more likely to find a better policy. A simulation is performed to demonstrate this with the mountain-car task. Figure 4-10 shows the number of steps to reach the goal, averaged over 10 independent runs by setting the shrinking rate to be near to zero

(as opposed to the original setting of $sr = 0.99$), such that the standard deviation of membership function will always be minimal. As shown in Figure 4-10, the best policy it can achieve is about 131 steps, which is much higher than the average 107 steps in Figure 4-7. This is because the system without the shrinking effect has a high bias, and less generalization. On the other hand, a possible explanation for the performance improvement in the system with a shrinking effect of the membership function is illustrated as follows. The system has started by grasping a “big picture” of the environment, and gradually converging towards better policies by improving on the accuracy. Therefore, the shrinking of the membership function during learning has a balanced bias-variance, and thus producing a 24 steps improvement in the learnt policy. Please see also Section 3.1.2 for more information on the bias-variance dilemma issue.

Figure 4-11: Cost-to-goal function of FITSK-RL.

Figure 4-11 shows the cost-to-goal function ($-\max_a Q(s,a)$) of the FITSK-RL model during a learning trial. The cost-to-goal function has its maximum value near the initial point of the simulation, and has its near-zero value at the goal position. It shows that the output of the FITSK-RL model is a smooth continuous function. It is easy to deduce that the output of FITSK-RL during any learning trial is similarly smooth due to the fuzzy interpolations of the local models. This is possible as long as the number of membership functions is adequately specified for each input dimension during design. This implies that FITSK-RL can effectively

model a continuous function output space, and provide good generalization to the value function estimations.

Figure 4-12: (a) Cost-to-goal, (b) action dynamics; during the 20000th trial.

Figure 4-12 shows the result of the simulation of FITSK-RL with a random start of initial state, during the 20000th trial of learning. Figure 4-12(a) shows the corresponding cost-to-goal function, and Figure 4-12(b) plots the action dynamics of the learnt-system (with 15x19 of sampling resolutions), the actions are full throttle reverse (black), zero throttle (grey), and full throttle forward (white). The possible optimal trajectories are hand-drawn and are illustrated in Figure 4-12 based on the learnt cost-to-goal function in Figure 4-12(a). The plotted optimal trajectories match with the action dynamics as shown in Figure 4-12(b), on the step for the optimal policy to take a forward action, and the time step to take a reverse action. The optimal policy is similar to the near-optimal policy illustrated in Figure 4-9. This shows that the learnt cost-to-goal function has accurately implied the optimal action to be taken at each possible state in the state space. This simulation demonstrates that FITSK-RL with SARSA learning is effective in the generalization of a continuous state, discrete action reinforcement learning problem.

Subsequently, a greedy policy is employed as the estimation policy in the FITSK-RL model, as described in equation (4-14). This formulation modifies the model into a Q-learning reinforcement learning model. The resultant FITSK-RL model with Q-learning is an off-policy method; the behaviour policy generates non-greedy action to control the car, and the estimation policy generates and improves on the current estimation.

Figure 4-13: Average number of steps to reach goal (FITSK-RL with Q-learning).

Figure 4-13 plots the number of steps to reach the goal over 2000 trials, averaged over 10 independent runs, for the resultant FITSK-RL model with Q-learning. The strength of the Q-learning is its early convergence comparing against SARSA learning illustrated in Figure 4-7. This is demonstrated by comparing Figure 4-13 with Figure 4-7. Figure 4-7 shows that the SARSA-learning process converged to the optimal policy after nearly 1650 trials of learning. However, the learning process for Q-learning converged after 830 learning trials. This is possible by estimating the action-value functions independent of the actual policy being followed, and enabling the early convergence of the greedy estimation policy. This simulation demonstrates that the FITSK-RL framework is able to assume the temporal learning of either SARSA or Q-learning algorithm with ease, without major model restructuring. This simulation demonstrates the generality of the reinforcement learning framework proposed in Section 4.1.1. The policy generator module of the generic framework can easily incorporate different policies, diversifying into different reinforcement learning algorithms.

Continuous state, continuous action FITSK-RL model:

The mountain-car task is modified to assume a single real-valued continuous action, where $a_t \in [-1, 1]$. The velocity and position dynamics are identical to the case of discrete action, as shown in equation (4-31). The single continuous action FITSK-RL model as discussed in Section 4.2.3 is employed for this simulation. The continuous action FITSK-RL model is a 3-inputs, single-output system. The corresponding 3-inputs is a vector of the state-action pair, $(s_t, a_t) = (x_t, v_t, a_t)$. The numbers of input linguistic nodes are 15, 15, and 3 respectively for the position, velocity, and action dimensions. This simulation adopts an on-policy SARSA

learning model, which has the ϵ -greedy action-selection strategy as the behavior and estimation policies. The maximum number of steps is limited to 300 steps.

Figure 4-14: Average number of steps to reach goal (continuous action FITSK-RL).

Figure 4-14 shows the simulation results averaged over 10 independent runs. There are 2500 trials for each run, the 1st 1500 trials started from any possible random state, and the next 1000 trials started from a fixed initial state, $s_0 = (-0.5, 0.0)$. As reported in [Smart 2000], the average number of steps to reach goal for near-optimal policy is 56 steps, for the learning with random initial state. Therefore, the 56 steps policy serves as the benchmarking baseline for the simulations. The continuous action HEDGER algorithm with Q-learning proposed in [Smart 2000] is able to achieve a performance of about 65 steps to reach the goal, without supplying initial training examples. The average number of steps to reach the goal for the continuous action FITSK-RL model is similar to the HEDGER algorithm, which is 64.45 steps. This is computed by averaging the performance from the 1001st trials to the 1500th trials. Therefore, the performance of the continuous action FITSK-RL is similar to the HEDGER algorithm.

Subsequently, the learning trials of continuous action FITSK-RL are continued with a fixed initial starting point from the bottom of the hill. The number of steps to reach the goal in fixed initial state simulation is expected to be larger than the previous simulation (random initial state), this is because the initial state is fixed to be further apart from the goal state. The result is plotted from the 1501st to the 2500th trials in Figure 4-14. The converged result is about 143.57 steps, which is averaged over the results from the 1701st to the 2500th trials. However, the best-averaged result for the discrete action FITSK-RL model is about 107 steps as shown in Figure 4-7, which is much better than the continuous action FITSK-RL model.

This demonstrates that the learning for continuous action model is harder than the case of discrete action reinforcement learning problem in terms of finding the optimal policy. This is because the continuous action model has a larger searching space, which is prone to be trapped in sub-optimal local maxima. This simulation demonstrates that the continuous action FITSK-RL model has similar performance with the HEDGER algorithm in solving continuous action, continuous state mountain-car task.

The simulations presented in this section demonstrated three properties of the FITSK-RL model, namely: a) The FITSK-RL model is applicable to three classes of reinforcement learning problems; these include discrete state-action (with puddle world task), continuous state-discrete action (with discrete action mountain-car task) and continuous state-action (with continuous action mountain-car task) tasks. b) The generic reinforcement learning framework proposed in Section 4.1 has good generality. The framework can be readily extended to SARSA learning or Q-learning without affecting the operations of other modules. c) The value generalization capability using the zero-order FITSK model for function approximation of reinforcement learning is good. The resultant FITSK-RL model has good generalization using less memory resources, as compared against the tile-coding model for generalization. Furthermore, the bias-variance of the system is being balanced with a simple setting to control the standard deviation of the membership function.

4.4 Summary Remarks

In this section, a generic reinforcement learning framework is proposed. The generic reinforcement learning framework employs a modular design with three modules: a) function approximator; b) policy generator; c) TD generator. A zero-order FITSK model is subsequently incorporated into the function approximator module of the generic framework. The resultant model is named the FITSK-RL model. The generic framework provides functionality abstraction to the modules, and improves the expandability of the functions in the FITSK-RL model.

The key strengths of the generic reinforcement learning framework are its modularity and generality. The modularity provides functional independencies among the modules, such that the reformulation of a single module would not affect the operations of the other modules. The generality applies to the generalization of different popular reinforcement learning

algorithms; namely: TD learning, SARSA learning and Q-learning, are mappable with ease into such a framework.

Two motivations drive the adoption of the zero-order FITSK model as the function approximator of the generic framework. Firstly, the problem domain is extended to continuous state-space. Secondly, the fuzzy operations of the FITSK model are defined by the TSK fuzzy inference scheme. This ensures that the operations of the function approximator is systematically defined, and has a strong theoretical foundation from the fuzzy inference scheme perspective. Subsequently, the FITSK-RL model is formulated to incorporate the approximations of $TD(\lambda)$, Q-learning, or SARSA learning with ease under a generic framework. Furthermore, the FITSK-RL model is extended to the continuous action domain by engaging a variable optimization method. This ensures that the FITSK-RL model is applicable to general classes of reinforcement learning problems, such as the discrete state-action, continuous state-discrete action, and continuous state-action domains.

The performance of the FITSK-RL model has been evaluated using two benchmarking simulations; namely: a) puddle-world task; b) mountain-car task. The results of the FITSK-RL model are encouraging when compared against other models. The memory resources of the FITSK-RL model are lesser than the tile-coding model [Sutton 1996b]. The generalization capability of the FITSK-RL is good, and can be further improved by having a control over the bias-variance of the system. The generic framework of the FITSK-RL model can readily be instantiated with SARSA learning and Q-learning. Finally, the performance of the FITSK-RL model is demonstrated to be similar to the HEDGER algorithm [Smart 2000] for a continuous action task. The key strength of the FITSK-RL is its generic capability, which allows it to be readily extensible to different temporal learning algorithms, and applicable for different discrete or continuous space problem domains.

This section summarizes the work on function approximation of reinforcement learning using the FITSK model. The next section will explore a possible extension of the temporal credit assignment problem into the *non-cumulative rewards* context of reinforcement learning. Subsequently, the non-cumulative rewards temporal learning algorithm will be incorporated into the generic reinforcement learning framework proposed in Section 4.1. An application of an optimal stopping problem will be demonstrated with the non-cumulative rewards FITSK-RL model in Section 5.

5 Maximum Reward Reinforcement Learning: A Non-Cumulative Reward Criterion

*"When I let go of what I am, I become what I might be."
~ Lao Tzu*

Reinforcement learning is a trial and error learning method, where the agent tries to explore and maximize the numerical state values from the feedback it receives over the long run. The process that the agent follows is modeled as the *Markov decision process* (MDP). In the MDP context, the two most well-studied optimality criteria are the expected cumulative reward and the average reward criteria [Puterman 1994]. The expected cumulative reward criterion has been adopted as the de facto objective measure in reinforcement learning research [Sutton 1984], [Sutton 1988], [Watkins 1989], [Rummery 1994], [Kaelbling 1996], [Bertsekas 1996], [Sutton 1998]. This criterion employs a technique to exponentially discount the future rewards, such that the near-future rewards are more valuable than far-future rewards. However, some research has demonstrated that the expected cumulative reward criterion is unsuitable for some problems, particularly in the undiscounted reward domain [Schwartz 1993]. Some reinforcement learning methods based on the average reward criterion have been proposed to resolve the undiscounted reward reinforcement learning problem [Schwartz 1993], [Mahadevan 1996], [Tadepalli 1998]. In the essence of this, both classes of methods employ some form of cumulative rewards criteria; the former with a reward discounting approach such that their infinite cumulative rewards are finite, and the latter with a reward averaging approach such that both short and long term rewards are indistinguishable.

However, a central issue is a problem of whether such a cumulative rewards formulation is a definite necessary requirement in all learning contexts. Is there any problem that the cumulative rewards formulation is unnecessary, and unjustifiable? This section discusses two possible scenarios where the cumulative reward formulation is to be avoided for superior learning result. This is demonstrated in Sections 5.2 and 5.3. In Section 5.2, a two-cycle task from [Mahadevan 1996] is discussed. One of the cycles leads to a sub-optimal policy and the other cycle leads to an optimal policy, but the rewards of the optimal policy is further away. Therefore, the focus of the learning is on how to bypass the sub-optimal policy in the search for an optimal policy. The classical Q-learning [Watkins 1989] was demonstrated to suffer from this problem because it was being dragged down by the sub-optimal policy, causing it to be lagging behind from switching to a better policy. In Section 5.3, an optimal stopping problem arising from financial derivative pricing [Tsitsiklis 1999] is discussed. This is a case

where the cumulative rewards formulation is unjustifiable. On the other hand, this problem is concerned with finding the supremum of the payoff values of each state, which is a function of the expected discounted maximum rewards in a non-cumulative manner. Furthermore, a discounted reward is significant in this domain as the discounting operation represents a compensation of future rewards with respect to the drift rate of financial pricing. This represents a contrary argument against the claim by [Schwartz 1993] that discounting of rewards is not necessarily important. Having these, a reinforcement learning formulation with non-cumulative *maximum reward* criterion is proposed to accommodate the learning needs of such tasks.

The maximum reward reinforcement learning agent exhibits a maximum reward-oriented behavior. It attempts to search for the policy that leads to the largest reward, irregardless of intermediate lower rewards along the path that lead towards the largest reward. This positions it to have superior immunity to the existence of sub-optimal policies in the problem domain, such as the two cycles problem in Section 5.2. Furthermore, the maximum reward learning is demonstrated to be effective in approximating the future payoff in a financial pricing context. The learning system exhibits an asymptotic behavior of approximating the supremum value with the maximum reward formulation, as will be discussed in Section 5.3.

This section is organized as follows. Section 5.1 describes the general formulation of the maximum reward reinforcement learning and its update formulae as part of the modular design of the earlier proposed generic reinforcement learning framework in Section 4.1. Section 5.2 studies the learning behavior of the maximum reward reinforcement learning using a two-cycle task, and compares against two existing reinforcement learning models; which are Q-learning and R-learning. Section 5.3 applies the proposed maximum reward reinforcement learning with its generic learning framework to an optimal stopping problem arising from the financial derivative pricing context. Section 5.4 concludes this section.

5.1 Maximum Reward Reinforcement Learning

This subsection formulates the maximum reward reinforcement learning. Section 5.1.1 mathematically formulates the maximum reward reinforcement learning, discusses the three existing reinforcement learning models, and the equivalent form of these three models for the maximum reward reinforcement learning. Section 5.1.2 solves the modelling problem of the maximum reward reinforcement learning. This is made possible by mapping the maximum

reward error update formula to a generic reinforcement learning framework to inherit the modularity and functionality of the framework.

5.1.1 Formulation of Maximum Reward Reinforcement Learning

This section defines a possible variation of return value, and extends the definition to various Bellman's optimality formulae from Section 2.1.1. This is followed by the formulation of the maximum reward form of temporal difference learning models in reinforcement learning.

The return value is defined as the cumulative future rewards in equation (2-2) where the agent is learning to maximize the return value. However, if the objective of the agent is to identify the largest reward within the value space, then the return value formulae in equation (2-2) can be suitably modified into equation (5-1).

$$R_t = \max(r_{t+1}, \gamma r_{t+2}, \gamma^2 r_{t+3}, \dots) = \max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+1}) \quad (5-1)$$

The maximum reward agent will attempt to discover the shortest path that lead to the highest reward, without concerning too much on the intermediate lower rewards. This constructs the fundamental behavior exhibited in the maximum reward reinforcement learning.

Given any state, s , and action, a , the probability of each possible next state, s' , or the *transition probability* from state s to state s' is given by equation (5-2).

$$P_{ss'}^a = \Pr(s_{t+1} = s' | s_t = s, a_t = a) \quad (5-2)$$

Similarly, the expected value of the next reward is given by equation (5-3).

$$R_{ss'}^a = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s') \quad (5-3)$$

where $E(.)$ is the expectation function.

Given a policy, π , which maps state s to an action a , the *value* of the state s under policy π , denoted by $V^\pi(s)$, is defined as the expected *return* at state s by following the same policy in all subsequent states. This is described in equation (5-4).

$$\begin{aligned}
 V^\pi(s) &= E_\pi(R_t | s_t = s) \\
 &= E_\pi(\max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+1}) | s_t = s) \\
 &= E_\pi(\max(r_{t+1}, \gamma \max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+2})) | s_t = s) \\
 &= \sum_{\forall a \in A(s)} \pi(s, a) \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma V^\pi(s'))]
 \end{aligned} \tag{5-4}$$

where $E_\pi(\cdot)$ is the expectation function with a given policy π .

Similarly, the *action-value function for policy π* is defined as the value of taking action a in state s under a policy π , denoted as $Q^\pi(s, a)$, and described by equation (5-5).

$$\begin{aligned}
 Q^\pi(s, a) &= E_\pi(R_t | s_t = s, a_t = a) \\
 &= E_\pi(\max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+1}) | s_t = s, a_t = a) \\
 &= E_\pi(\max(r_{t+1}, \gamma \max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+2})) | s_t = s, a_t = a) \\
 &= \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma V^\pi(s'))] \\
 &= \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma \sum_{\forall a' \in A'(s')} \pi(s', a') Q^\pi(s', a'))]
 \end{aligned} \tag{5-5}$$

Consider an *optimal policy*, π^* , where,

$$V^{\pi^*}(s) \geq V^\pi(s), \text{ for all } s \text{ and } \pi. \tag{5-6}$$

An optimal policy produces the largest state value for all s among all the policies. Define the *optimal state-value function*, V^* , as equation (5-7).

$$\begin{aligned}
 V^*(s) &= \max_{\pi} V^\pi(s) \\
 &= \max_a Q^{\pi^*}(s, a) \\
 &= \max_a \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma V^*(s'))]
 \end{aligned} \tag{5-7}$$

Similarly, the *optimal action-value function*, Q^* , is described by equation (5-8).

$$\begin{aligned}
 Q^*(s, a) &= \max_{\pi} Q^\pi(s, a) \\
 &= E(\max(r_{t+1}, \gamma V^*(s_{t+1})) | s_t = s, a_t = a) \\
 &= E(\max(r_{t+1}, \gamma \max_{a'} Q^*(s_{t+1}, a')) | s_t = s, a_t = a) \\
 &= \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma \max_{a'} Q^*(s', a'))]
 \end{aligned} \tag{5-8}$$

Equations (5-7) and (5-8) represent the maximum reward Bellman optimality equations for V^* and Q^* .

The temporal difference error equations for the three popular reinforcement learning models are listed in equations (5-10), (5-11) and (5-12). These models are the temporal difference learning [Sutton 1988], [Sutton 1998], the SARSA learning [Rummery 1994], and the Q-learning [Watkins 1989] respectively.

The general TD learning error follows from the error function in equation (5-9).

$$\delta_t = R_t - E(R_t) \quad (5-9)$$

Therefore, the TD error of temporal difference learning for a state at time t is given in equation (5-10).

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \quad (5-10)$$

For on-policy SARSA learning, the TD error is described in equation (5-11).

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \quad (5-11)$$

Similarly, for off-policy Q-learning, the TD error is described in equation (5-12).

$$\delta_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \quad (5-12)$$

The three popular reinforcement learning models are modified into a respective form of maximum reward reinforcement learning model. It can be formulated by modifying the return value to take on the maximum reward, as shown in equation (5-1). Following these, the TD error formulae for maximum reward reinforcement learning models are shown in equation (5-13). The subscript M denotes that the objectives of these learning models are the maximum reward values instead of the cumulative reward values.

$$\begin{aligned} \text{TD}_M \text{ learning: } \delta_t &= \max(r_{t+1}, \gamma V_t(s_{t+1})) - V_t(s_t) \\ \text{SARSA}_M \text{ learning: } \delta_t &= \max(r_{t+1}, \gamma Q_t(s_{t+1}, a_{t+1})) - Q_t(s_t, a_t) \\ \text{Q}_M \text{-learning: } \delta_t &= \max(r_{t+1}, \gamma \max_a Q_t(s_{t+1}, a)) - Q_t(s_t, a_t) \end{aligned} \quad (5-13)$$

Further extensions of these maximum reward TD error formulae to the respective reinforcement learning models are straightforward and will be discussed in Section 5.1.2. The

differences in the behaviour and the strengths of the maximum reward reinforcement learning over existing reinforcement learning models will be demonstrated with simulations and examples in the latter part of this section.

5.1.2 Mapping to a Generic Reinforcement Learning Framework

This subsection extends the maximum reward reinforcement learning update formulae (equation (5-13)) into the generic reinforcement learning framework proposed earlier in Section 4.1 and in [Quah 2004c]. The block diagram of the generic agent of the framework is shown in Figure 4-1 of Section 4.1.

The generic agent adopts a modular design with three modules; namely: the function approximator, the policy generator, and the temporal difference (TD) generator. The function approximator module predicts the value functions based on the state information. The policy generator module generates an appropriate action and the corresponding action-value. The TD generator module subsequently formulates the temporal difference error based on the reinforcement signal, the current action-value, and the successive action-value. Consequently, the temporal difference error is feedback to the function approximator module to adjust the tunable parameters. Two motivations for having this modular design are the functional generality or abstraction, and the functional independence. Having generic functionality for each respective module enables the framework to be abstractly defined, and thus easily extendable to different reinforcement learning model such as TD learning, SARSA learning, or Q-learning. Moreover, the functional independency ensures that the modules are not tightly coupled on each other, and thus reformulation or implementation of a single module would not affect the general operations of the other modules. The discussion in this section is limited to the following extent; the TD generator module is being generalized to incorporate the learning of maximum reward update formulae. Please refer to Section 4.1 on the definitions of the function approximator and the policy generator modules.

The TD generator computes the temporal difference error as a function of the immediate reward, current action-value, and successive action-value estimation, as shown in equation (5-14).

$$\text{TD error: } \delta_t = f_{TD}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) \quad (5-14)$$

where $f_{TD}(\cdot)$ is a generic function of the TD generator module.

The generic TD function is extended to the TD error of the cumulative reward formulation with equation (5-15).

Cumulative reward TD error:

$$\delta_t = f_{TD}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) = r_{t+1} + \gamma Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \quad (5-15)$$

Similarly for the maximum reward formulation, the generic TD function adopts the formula as shown in equation (5-16).

Non-cumulative (maximum) reward TD error:

$$\delta_t = f_{TD}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) = \max(r_{t+1}, \gamma Q_t(s_{t+1}, a')) - Q_t(s_t, a_t) \quad (5-16)$$

The TD error formulae in equations (5-15) or (5-16) can be easily mapped to TD learning, SARSA learning or Q-learning by adopting different action selection strategies in the policy generator module, as demonstrated in Section 4.1. Furthermore, the maximum reward reinforcement learning can be directly incorporated into the FITSK-RL model by adopting a zero-order FITSK model [Quah 2004b] from Section 3.1.2 as the function approximator module. This enables the value function to be effectively approximated with a linear approximator (linear to the tunable parameter) in a systematic manner, and extendable to learning with the eligibility trace formulations [Sutton 1984]. Please refer to Section 4 on the technical details of formulating the FITSK-RL model.

5.2 Comparison with Q-learning and R-learning

This section compares the proposed non-cumulative, maximum reward reinforcement learning with existing learning models with inherent cumulative rewards reinforcement learning models, particularly the Q-learning [Watkins 1989] and the R-learning [Schwartz 1993]. Q-learning is an off-policy, model-free reinforcement learning model, and is shown to be effective in control task learning with discounting of cumulative rewards. The learning equation for Q-learning is described by equation (5-12). The basic idea of formulating a Q-learning control is to adopt a greedy policy as the estimation policy, while simultaneously exploring the search space with a non-greedy behaviour policy. Similarly, R-learning is an off-policy, model-free model. However, the objective of R-learning is to maximize the performance based on the undiscounted cumulative rewards value, instead of the discounted cumulative rewards value described in equation (2-2). The undiscounted cumulative reward value of a process starting at time t is shown in equation (5-17).

$$\rho_t = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^n r_{t+k+1} \quad (5-17)$$

where ρ_t is the average undiscounted cumulative reward at time t . It was shown in [Schwartz 1993] that if a policy gives rise to an *ergodic* Markov chain, the average reward values for all the states in that ergodic Markov chain are identical (equation (5-18)).

$$\rho = \rho_t, \forall s_t \in S_E \quad (5-18)$$

where S_E is the set of ergodic states.

The update formulae of R-learning are represented in equation (5-19). Please refer to [Schwartz 1993] on the rational of these update formulae.

$$\begin{aligned} RV_{t+1}(s_t, a_t) &= RV_t(s_t, a_t) + \alpha_1(r_{t+1} - \rho_t + \max_a RV_t(s_{t+1}, a) - RV_t(s_t, a_t)) \\ \rho_{t+1} &= \rho_t + \alpha_2 G(a_t)(r_{t+1} + \max_a RV_t(s_{t+1}, a) - RV_t(s_t, a_t) - \rho_t) \end{aligned} \quad (5-19)$$

where,

$RV(.)$ is a vector of real numbers, corresponding to each state-action pair (s, a) ,

α_1, α_2 are learning rate parameters, and

$G(a_t)$ is a logic function with output 1 if a_t is greedily selected, and 0 otherwise.

Figure 5-1 shows an example from [Mahadevan 1996], which is an adaptation of the two cycles Markov decision process (MDP) task [Schwartz 1993] with action choice at state 1 (Home) of the MDP. A robot following that MDP receives reward of +5 at the Printer room (state 14) for a trip from the Home to the Printer room, and receives reward of +20 for servicing the Mailroom (state 10) that is further away. Learning simulations are performed for R-learning (equation (5-19)), Q-learning (equation (5-12)) and Q_M -learning (equation (5-13)) respectively. All the learning rates have numerical value 0.2, and all the behavior policies are ϵ -greedy [Sutton 1998] with $\epsilon=0.05$.

Figure 5-1: An MDP with two cycles.

Figure 5-2: Comparison of R-learning, Q-learning and Q_M -learning.

Figure 5-2 shows the performance comparison between R-learning, Q-learning and Q_M -learning. The results are averaged over 5 independent runs, each run consisted of 20000 number of steps. The performance is measured with the average of rewards being accumulated over the past 1000 number of steps. The optimal path follows the cycle from 1-2-3-4-5-6-7-8-9-10, and repeats itself infinitely. Therefore, the maximum of the average reward value is 2, and is derived by averaging the largest reward, 20, over 10 actions for each cycle. The learning agent may often stay at the suboptimal policy, which follows the cycle from 1-11-12-13-14, prior to arriving at the optimal policy. This suboptimal policy has an average reward of 1. A discount rate, $\gamma=0.9$, is adopted for both the Q-learning and the Q_M -learning. The discount rate in this MDP has to be larger than 0.7579 such that the policy of servicing the mailroom is being chosen. Any discount rate less than 0.7579 will cause a larger value at state 11 than the value at state 2. Thus, the policy servicing the Printer room will look more attractive than the policy servicing the Mailroom. R-learning arrived at the optimal solution in about 3700 number of steps, whereas Q_M -learning and Q-learning took 10700 and 13500 steps respectively to arrive at the optimal solution. Comparing with R-learning in the undiscounted formulation, the learning with discounted formulation is prone to convergence to the suboptimal solutions. However, the maximum reward Q_M -learning performs slightly better than the cumulative reward Q-learning. The reason pertaining to this phenomenon will be discussed in the next simulation.

Figure 5-3: Comparison of R-learning, Q-learning and Q_M -learning with eligibility traces.

The reason that R-learning outperforms Q_M -learning in the previous simulation is investigated in Figure 5-3. Eligibility trace formulations [Sutton 1984], [Sutton 1998] are introduced to the Q-learning and Q_M -learning with a large trace-decay parameter, $\lambda=0.9$. This ensures that the future rewards are apparent to the learning agent as soon as they are being discovered by a non-greedy exploratory policy. Figure 5-3 shows that the Q_M -learning with eligibility trace has superior performance to the others. The Q_M -learning is more far-sighted than the others, by adopting the eligibility trace. Q_M -learning converges in about 1500 number of steps, while Q-learning and R-learning converges in about 3700 number of steps. The Q_M -learning converges to the optimal policy in the shortest possible timeframe without being dragged down by the suboptimal policy. This is consistent with the behaviour of maximum reward agent introduced in Section 5.1.1. The agent will attempt to discover the shortest path that leads to the highest reward, without concerning too much on the intermediate lower rewards. However, for cumulative rewards models such as Q-learning or R-learning, it takes time for them to fade away the influence of the lower reward and the effect of the suboptimal policy. Therefore, the convergence of these cumulative rewards models is not as immediate as the non-cumulative maximum reward model.

5.3 Optimal Stopping Problem in Financial Derivative Pricing

In the original paper on average reward reinforcement learning [Schwartz 1993], the author claimed that one of the reasons for discounting rewards is to compensate for the interest rate

of future rewards, especially in the field of economics study. This has been demonstrated by [Van Roy 1998], [Tsitsiklis 1999] in the studies of an optimal stopping problem in financial derivative pricing. However, the problem cannot be readily resolved by the discounted accumulative reward reinforcement learning model. This is because the objective of the problem is to determine the optimal stopping state with the highest discounted non-cumulative rewards. This section attempts to demonstrate the application of the maximum reward reinforcement learning to successfully model this financial derivative pricing problem.

5.3.1 Problem Formulation

The optimal stopping problem was originally formulated by [Van Roy 1998], [Tsitsiklis 1999] for optimal pricing prediction of an exotic derivative product [Parsley 1997]. The problem domain is a contract based on the daily prices of a single stock. When the contract is exercised, a payoff is received in an amount equal to the current price of the stock divided by the price a 100 days before. Please refer to [Tsitsiklis 1999] on the background studies and details of this problem. A summary of this problem is given as follows.

The process of stock prices is generated based on a *geometric Brownian motion* model [Hull 1997]. The change in the stock price is modelled as a normal distribution with *drift rate* as the mean, and with *volatility* as the standard deviation, as described in equation (5-20).

$$\frac{\Delta p_t}{p_t} \sim \phi(\mu\Delta t, \sigma\sqrt{\Delta t}) \quad (5-20)$$

where,

- p_t is the stock price at time t ,
- μ is the expected drift rate (percentage return, or interest rate of the stock price),
- σ is the volatility (standard deviation of the stock price),
- Δt is the interval of time, and
- $\phi(m,s)$ is a normal distribution with mean m and standard deviation s .

If the prices are computed on a daily basis, then the time interval is 1, and equation (5-20) is simplified into the update equation described in equation (5-21).

$$p_{t+1} = p_t(1 + \mu + \sigma\phi(0,1)) \quad (5-21)$$

This problem considers the daily volatility of 0.02 and annual interest rate of 10%; the drift rate is 0.0004, this is computed by dividing the interest rate by the number of market opening

days (~250 days) [Hull 1997]. Therefore, the daily prices are updated based on equation (5-22).

$$P_{t+1} = P_t(1.0004 + 0.02\phi(0,1)) \quad (5-22)$$

where $t \geq -100$.

The information available to the learning agent at a particular discrete time t is a vector, described as follows,

$$x_t = \left(\frac{P_{t-99}}{P_{t-100}}, \frac{P_{t-98}}{P_{t-100}}, \dots, \frac{P_t}{P_{t-100}} \right), t \geq 0$$

The payoff is rewarded if the contract is terminated at time t . Therefore, the reward received a time step later is the last component of the information vector, $r_{t+1} = x_t(100)$. Then the value of the contract at time t is given by equation (5-23).

$$V_t = \sup_{\forall k \geq 0} E(\gamma^k r_{t+k+1}), t \geq 0 \quad (5-23)$$

and the discount rate is given by, $\gamma = e^{-\mu}$. If t^* is an optimal stopping time, then the optimal value of the contract is described by equation (5-24).

$$V_{t^*} = E(r_{t^*+1}) = \sup_{\forall k \geq 0} E(\gamma^k r_{t^*+k+1}) \quad (5-24)$$

5.3.2 Learning Model Formulation and Results

The state of the problem is constructed based on the following rationale. If there was a large slump and recovery in the stock prices within the past 100 days, then probably the contract should be held for some time to reap potential future benefits [Van Roy 1998], [Tsitsiklis 1999]. Therefore, the most important information is dependant on the current stock price, and the minimum stock price over the past 100 days. With this rationale, the state information is constructed as shown in equation (5-25). The first state dimension represents the proportional increment of the current stock price. The second state dimension implies the difference between the proportional increment of the current stock price over the minimum stock price of the past 100 days.

$$s_t = \left(\frac{P_t}{P_{t-100}}, \frac{P_t}{\min_{(t-100) \leq \tau \leq t} P_\tau} - \frac{P_t}{P_{t-100}} \right), s_t \in \mathbb{R}^2 \quad (5-25)$$

In simulation trials, since the number of training iterations is always finite, there exists a maximum value in the set of the expected discounted payoff, and this maximum value is also the supremum of the set (equation (5-26)).

$$V_t = \max_{\forall k \geq 0} E(\gamma^k r_{t+k+1}) = \sup_{\forall k \geq 0} E(\gamma^k r_{t+k+1}), t \geq 0, k \text{ is finite} \tag{5-26}$$

Equation (5-26) is equivalent to equation (5-4), which is the value function of the maximum reward reinforcement learning. Thus, this optimal stopping problem can be approximated with the maximum reward reinforcement learning, TD_M-learning, where the update error formula is shown in equation (5-13).

In this simulation, the optimal value function is approximated with the FITSK-RL model. The FITSK-RL model consists of a zero-order FITSK as the function approximator. The parameters for the learning are shown in Table 5-1.

Table 5-1: Predefined FITSK-RL parameters

Learning constant, $\alpha = 0.001$	Discount rate parameter, $\gamma = e^{-\mu} = e^{-0.0004}$
Dimensions = 2	Trace-decay parameter, $\lambda = 0.5$

The first and second dimensions are denoted as $s(1)$ and $s(2)$, which are computed using equation (5-25). The input ranges for the two state dimensions are $s(1) \in [0.5, 2.0]$ and $s(2) \in [0.0, 0.6]$ respectively. The number of pre-assigned membership functions for the two dimensions are 5 and 3. The total number of parameters to tune are $5 \times 3 = 15$. All the initial tunable parameters are set to zeros. The learning is iterated for 500,000,000 times. The approximated value function from the output of the model is plotted in Figure 5-4.

Figure 5-4: Approximated optimal value function with maximum reward FITSK-RL.

Figure 5-4 shows the surface plot of the approximated optimal value function with the proposed maximum reward FITSK-RL model after the learning iterations. The two state dimensions are represented with the X-Y axes, the approximated optimal value function is represented with the Z axis. When the current stock price is near to the bottom of the past 100 prices ($s(2) \rightarrow 0$), then there is not much chance of getting better further benefits, and therefore expecting lower payoff. When the current stock price yields a growth of about 1.2 in $s(1)$, the expectation of payoff increases proportionally with $s(2)$. Increased in $s(2)$ means that there is a large slump and recovery over the past 100 days, and probably one should wait for sometime to reap potential higher payoff. The region at the upper right corner ($s(1) \rightarrow 2.0$, $s(2) \rightarrow 0.6$) is rarely encountered (see Figure 5-4), and thus their learnt optimal value is lower than expected due to the lack of training examples. However, when such situation occurs, the process is terminated immediately to reap the high benefits from these states. This is because the chance of getting any potential higher payoff than the current is quite slim.

Denoting the approximated optimal value function from the FITSK-RL model as $\tilde{V}(s_t)$, then the stopping time, $\tilde{\tau}$, is computed by equation (5-27). This is similar to the terminating criterion defined by [Tsitsiklis 1999], which was proven to be the optimal stopping time.

$$\tilde{\tau} = \min\{t \mid r_{t+1} \geq \tilde{V}(s_t)\} \tag{5-27}$$

Table 5-2: Results of the optimal stopping problem.

Strategy	Expected reward	Standard deviation	State dimension
Thresholding	1.238	-	-
Tsitsiklis model	1.282	0.0022	100
FITSK-RL	1.289	0.0029	2

The expected reward corresponding to the approximated optimal value function is estimated by averaging the rewards gathered over 50000 independent trajectories, and terminated according to equation (5-27). The results are tabulated in Table 5-2. The expected reward computed from such stopping criterion is 1.289127, and the estimated standard deviation for this sample mean is 0.002856. The expected reward is 1.238 from a thresholding strategy; and 1.282 (with 0.0022 standard deviation for sample mean) for the approximated optimal value function from [Tsitsiklis 1999] formulation. Therefore, the expected payoff from the maximum reward FITSK-RL model is slightly better than the formulation by Tsitsiklis, which is leading by 0.007127. The expected reward is numerically 0.051127 better than the thresholding strategy. This means that the optimal pricing of the strategy identified by the

maximum reward FITSK-RL model is 5% better than the thresholding strategy, and is about 1% better than the Tsitsiklis formulation.

5.3.3 Discussion

In the original paper on average reward reinforcement learning for the undiscounted reward problem, the author claimed that using discounting of rewards to compensate for the interest rate is unjustifiable [Schwartz 1993]. However, the optimal stopping problem in financial derivative pricing studied in this subsection shows the contrary. The rewards are discounted by $e^{-\mu}$ to compensate for the possible future increment of the contract values, such that future values are comparable side-by-side with the current values. On the other hand, discounted cumulative reward reinforcement learning is not applicable to this problem. This is because the objective of the problem is to search for a highest expected reward for each state. Having this, a non-cumulative, discounted, maximum reward reinforcement learning formulation is shown to be directly applicable in solving this problem.

The FITSK-RL model is demonstrated to incorporate the maximum reward reinforcement learning formulation with ease in Section 5.1.2. The problem in this section is simulated with the maximum reward FITSK-RL model. The formulation presented in this section is slightly different from the one proposed in [Tsitsiklis 1999]. They formulated the problem with 100 state dimensions to demonstrate the good scaling of their local basis function approximator. However, the formulation in this section reduces the state dimension to 2, such that the learning result can be easily visualized and inspected (as shown in Figure 5-4). This improves the interpretability of the learning result. Furthermore, the operations of the FITSK-RL model are systematically defined, as opposed to the manually generated local basis function approximation method in [Tsitsiklis 1999]. The expected payoff of the maximum reward FITSK-RL model with 2 state dimensions is slightly better than the existing methods with 100 state dimensions. This shows that the state definition in equation (5-25) with two dimensions can adequately provide enough information to the learning system. This is a significant improvement in modelling complexity over the existing methods.

5.4 Summary Remarks

In this section, a new maximum reward criterion is proposed as the objective of reinforcement learning. The maximum reward criterion is subsequently incorporated into the reinforcement learning update formulae from the Markov decision process perspective. Subsequently, the maximum reward reinforcement learning is mapped to the generic

reinforcement learning framework to solve the modelling issue of this new learning paradigm.

The distinct characteristic of the maximum reward reinforcement learning is a non-cumulative, maximum reward-oriented behaviour of the learning agent, as opposed to the usual cumulative rewards-oriented behaviour. It attempts to search for the policy that leads to the largest reward, irregardless of intermediate lower rewards. This positions it to have superior immunity to the existence of sub-optimal policies. This is demonstrated with the example in Section 5.2, where the maximum reward agent established a convergence to the optimal policy in the shortest timeframe. However, other agents with underlying cumulative rewards-oriented behaviour are dragged down by the sub-optimal policy and lower reward. As a result, they suffer from slower convergence towards the optimal policy.

Section 5.3 demonstrates a successful application of maximum reward reinforcement learning to asymptotically approximate the supremum of payoff values in an optimal stopping problem. The maximum reward FITSK-RL model proposed in Section 5.1.2 has been adopted to resolve this problem. This problem cannot be effectively solved using the usual cumulative rewards reinforcement learning paradigm because the underlying payoff values are non-cumulative. Despite reducing the problem to a two-state dimension problem, the approximation result using the FITSK-RL model is slightly superior to the original formulation with 100 state dimensions in [Tsitsiklis 1999]. Furthermore, this improves the interpretability of the learning results and reduces the modelling complexity of the problem. The approximation result is visualized such that the rationale of the result can be examined with ease. This would not be possible with the original formulation of the problem as the state dimensions are too large and it is hard to gain insight into the learning result of the manually defined local basis functions. However, this is possible with the FITSK-RL model as the operations are systematically defined.

In summary, the maximum reward reinforcement learning approach is proposed as a possible new learning paradigm to exhibit superior performance for problems with underlying non-cumulative nature of reward values. Examples and simulations are provided to illustrate this. The modeling issue of the maximum reward reinforcement learning is solved by mapping it into the generic reinforcement learning framework. However, the theoretical research of maximum reward reinforcement learning is still at its infancy stage, particularly from the optimal control perspective. The convergence of maximum reward reinforcement learning

resulted from a linear function approximator is yet to be established. Some possible resemblance of the convergence result may possibly be extended from the Tsitsiklis' papers [Tsitsiklis 1997], [Tsitsiklis 1999]. The relationship to the sensitive discount optimality criterion [Puterman 1994], [Mahadevan 1996] has not been studied, and is worthy of a further research effort.

6 MCES: The Monte Carlo Evaluative Selection Approach

*"The strongest principle of growth lies in human choice."
~ George Eliot*

Feature subset selection (or feature selection, input selection) is a preprocessing step in computational learning task. It generates significant computational advantages by reducing the input dimensionality. Hence, this often results in better learning accuracy, and the improvement in the interpretability or the comprehensibility of the system by the judicious removal of irrelevant input features or attributes.

Feature subset selection has been extensively studied to improve the performance of some induction methods. These induction methods include Naive-Bayes [Duda 1973], decision tree-based induction algorithms (i.e., ID3 [Quinlan 1986], C4.5 [Quinlan 1993]), and CART [Breiman 1984]. The feature selection problem for such induction methods has been very well studied; some of the well-known methods are FOCUS [Almuallim 1991], [Almuallim 1994], Relief [Kira 1992a], [Kira 1992b], [Kononenko 1994], Cardie's decision tree algorithm [Cardie 1993], and several wrapper approaches [John 1994], [Kohavi 1997]. Good surveys on these algorithms have been undertaken by Blum and Langley [Langley 1994], [Blum 1997]. Some recent advances of feature selection algorithms are also reported in [Jang 1996], [Utans 1991], [Laar 1997], [Kwak 2002], [Mao 2002], [Setiono 1997], [Setiono 2002], [Lee 2001], [Hsu 2002], [Petersen 1998], [Shen 1999], [Shen 2001a], [Shen 2002], [Shen 2004], [Jensen 2005], [Fu 2003]. Some successful applications of feature selection on complex problems are also described in [Shen 2000], [Shen 2001b], [Jensen 2004], [Furlanello 2003] and [Cantu-Paz 2003]. Some of these approaches will be selectively reviewed and studied in Sections 6.1.2 and 6.1.3 of this Thesis.

These feature subset selection algorithms have drawn significant attention and have been widely adopted in the data mining communities. However, most of these algorithms suffer from one of the following limitations; namely:

- only applicable to the classification task domain; and
- tightly-coupled with a specific induction algorithm.

Most of the filter approaches assume some type of relevancy measures to determine the relevancy of the input features to the output target. These relevancy measures are derived

from information theoretic approaches and statistical analysis, which are efficient in the classification task domain. However, the regression problems are highly complex and the relevancy measures are not directly computable. Therefore, most filter and wrapper approaches designed to solve the feature selection problem of the classification task are not necessarily applicable to the regression problems. For example, the FOCUS algorithm [Almuallim 1991], [Almuallim 1994] is not applicable to select features for a system with nonlinear output functional space.

Wrapper approaches wrap around a specific induction algorithm in order to search for the best set of features guided by the learning or performance of the induction algorithm. In order to optimize the features set, most of the wrapper approaches assume some form of techniques specific only to the induction algorithm being adopted. For example, Setiono's method [Setiono 1997] is dependent on the neural network saliency measure, and Jang's method [Jang 1996] is dependent on the first epoch learning error of its ANFIS structure. Feature selection algorithms designed in this manner are not readily applicable to other induction algorithm. For example, Setiono's feature selection method cannot be used with ANFIS learning and Jang's feature selection method cannot be applied to neural network learning. This resulted in enormous effort in redesigning a feature selection algorithm when the induction algorithm or learning system is modified or changed for a particular task.

This section attempts to generalize the wrapper approach feature subset selection method to be independent of the application domain, and the underlying induction algorithm. As a wrapper approach, it can be easily re-deployed with other induction algorithms. The only constraint is that the accuracy measures are quantifiable from the induction algorithm, which is easy to meet. This is because accuracy measures are needed to guide the learning process of the induction algorithm itself, which is the prior process to feature selection.

The initial motivation of this section is extended from the discussion in Section 3.1.3 of this Thesis and is summarized as follows. The localized Kalman filter approach has been devised to relax the curse-of-dimensionality issue in parameter learning of the first order TSK model. The localized Kalman filter approach has been empirically shown to apply to problems with larger than 10 input dimensions. This is in contrast to the original Kalman filter approach, which is limited to about 6 input dimensions. Despite these, any first-order fuzzy model with high input dimensions (particularly up to 30 input dimensions), is still precariously applicable in practice with the TSK models. Furthermore, systems with a high input dimension may

defeat the purpose of using the TSK model for system estimation. This is because the model interpretability, which is a significant strength in the TSK model, is not apparent in high dimensionality systems. Therefore, contents in this section are devoted to solve the feature selection problem as an independent component orthogonal to the topics discussed in Sections 2 - 5. The proposed method in this section complements the research in the previous sections as well as acting as an independent model in solving some feature selection problems. The proposed method will select the input features, and limit the system dimensionality such that it is possible to comprehend the system interpretability of the TSK models. Having this, the reduced feature set will produce greater fidelity to the learning system without a loss of generality to the result.

Section 6.1 surveys some of the existing feature selection approaches that are applicable to regressive task. For a detailed survey of feature selection problem for classification task, please refer to the survey by [Blum 1997]. Section 6.2 discusses the design and implementation of the proposed algorithm, Monte Carlo Evaluative Selection (MCES). Section 6.3 presents extensive experimental results of the MCES approach in both classification and regressive learning domains. The strengths of MCES over other existing techniques will be highlighted using these findings. Section 6.4 summarizes this section.

6.1 Background Studies

Existing feature selection methods are reviewed in this subsection, with special emphasis on those that are applicable to regressive task. For those methods that are only applicable to classification task and are being widely adopted in data mining, readers are advised to refer to the surveys undertaken by Blum [Blum 1997], John [John 1994], and a book by Liu [Liu 1998].

6.1.1 Problem definition

With a lack of domain knowledge, the usual approach in any system modelling is to include all possible input variables that may have effects on the output. Furthermore, the number of tunable parameters in the system also increases in order to learn the complex input-output relationship [Laar 1997]. This leads to the following problems; namely:

- Increase in the number of redundant tunable parameters in the learning system, and thus degrading the generalization ability and accuracy of the system.
- Increase in the computational and space complexities with these redundant attributes.

- Degradation in the expressiveness of the system.

Two classes of features are to be excluded in a feature selection system; namely:

- Irrelevant features: Depending on the relevancy measures, features can be categorized as strongly relevant, weakly relevant, and irrelevant. In computational learning, accuracy is one of the efficient measures to define the relevancy. It is a relationship between the input features and the output target.
- Correlated features: These are distinct from the relevance of the features. If there are features that are highly correlated to some other features in the same system, then they can be considered as redundant features and should be removed. This is a relationship amongst the input features themselves.

The goal of the feature selection algorithm is to remove unneeded features to improve the performance of the system.

6.1.2 Regressive Task Feature Selection methods

Some of the existing feature selection methods with possible application to regressive tasks are discussed in this subsection.

(a) ANFIS Input Selection

This is a simple and practical input selection method, developed by Jang [Jang 1996]. The method works on his *Adaptive Neuro-Fuzzy Inference Systems* (ANFIS) architecture [Jang 1993]. The method is based on the implicit assumption that the ANFIS model having the smallest error after one epoch of training, has a greater potential of achieving a lower error when further epochs of training are involved. This assumption is not absolutely accurate but reasonable.

For example, if there are 10 inputs with a target to find the three most influential inputs, the method will exhaustively construct $C_3^{10} = 120$ ANFIS models, each with a different combination of the three inputs. Each model undergoes an epoch of training, and the combination of the input features that yields the smallest training error is subsequently selected.

The method has been tested using two applications; namely: the automobile MPG Prediction, and the Box and Jenkins gas furnace data [Jang 1996]. The automobile MPG Prediction

problem is used as one of the benchmark problem in Section 6.3; the result is subsequently contrasted against the ANFIS selection method.

This input selection method is deficient in three aspects; namely:

- The assumption to consider the error measures after the first epoch of training may not be accurate. Especially when some of the feature combinations have errors that are near to the optimal combination, then these combinations have a good potential of outperforming the target combination after a few epochs of training.
- This method is exhaustive and the number of input features to be selected has been predetermined.

(b) Setiono's Neural Network Feature Selector

Setiono has proposed two similar approaches in two papers [Setiono 1997], [Setiono 2002] for the feature selection in his neural network models. The main difference is in the definition of the error objective function of the neural network model. Their methods start with a complete set of input attributes, and subsequently prune the network input units based on an accuracy ranking.

Both methods monitor the decline in accuracy by setting all the weights connecting to a specific input feature to zeros; and remove the one that yields minimum decrement in accuracy. After each iteration, a feature is removed and the network is retrained with an updated set of features.

These methods work on both classification and regression tasks. However, some possible limitations are listed below; namely:

- The methods are tightly coupled with a particular neural network model and the error function. It is difficult to extend this approach to other learning architectures.
- The methods assume that those weights that are near-to-zero are redundant weights. This assumption is only valid if all the input instances have been normalized to mean values of zero, or there is some normalizing function in the input nodes (i.e. bipolar sigmoid function).
- Retraining of the neural network models is needed, and thus the methods are computationally expensive.

(c) Sensitivity-based Pruning (SBP)

Moody and Utans [Utans 1991], [Moody 1992], [Moody 1995] proposed a Sensitivity-based Pruning method (SBP) in neural network application. The algorithm computes a sensitivity measure to evaluate the changes in training error resulting from the removal of an input. The sensitivity is based on the training error between the original output and the output of the same network having a disabled input.

Assume that there are J input features, with I training instances, and,

- $\mathbf{X}$ is the matrix of input variables,
- $\mathbf{x}_j$ is the transpose vector of all exemplars for feature j , where $\mathbf{x}_j = [x_{1j} \ x_{2j} \ \dots \ x_{ij} \ \dots \ x_{Ij}]^T$,
- $\overline{x_j}$ is the average value of all elements in $\mathbf{x}_j$,
- $\mathbf{x}_i$ is the vector of all input features for exemplar i , where $\mathbf{x}_i = [x_{i1} \ x_{i2} \ \dots \ x_{ij} \ \dots \ x_{iJ}]$,
- x_{ij} is the element of feature j with exemplar i , $x_{ij} \in \mathbf{X}$,
- S_j is the sensitivity measure for feature j ,
- S_{ij} is the sensitivity measure of x_{ij} ,
- W is the weight vector of the neural network, and
- $SE(.)$ is the square error function.

The sensitivity measure of x_{ij} is computed with equation (6-1).

$$S_{ij} = SE(\overline{x_j}, W) - SE(x_{ij}, W) \quad (6-1)$$

The *Average Substitution* is performed by substituting an element in $\mathbf{x}_j$ with the average value of all elements in $\mathbf{x}_j$, as shown in equation (6-2).

$$\overline{x_j} = \frac{1}{I} \sum_{i=1}^I x_{ij} \quad (6-2)$$

The sensitivity measure of feature j is subsequently computed with equation (6-3).

$$S_j = \frac{1}{I} \sum_{i=1}^I S_{ij} \quad (6-3)$$

The sensitivity measure is the degree of relevancy of that particular input feature to the target output. The algorithm virtually disables the influence of an input feature by substituting a variable by its average value (equation (6-2)). It is insufficient merely to disable the input by setting it to zero. This is because the bias of each hidden unit is determined during learning and cannot be properly offset by simply setting the input arbitrarily to zero.

The strength of this method is that it does not involve the retraining stage, it evaluates the square error based on the current weights of the network, and thus the computational cost is lower. Since this method evaluates the accuracy based on the training error, and does not involve changes of weights, it can be easily extended to other induction architectures.

However, four possible weaknesses are identified in this algorithm; namely:

- The algorithm neglects the possible correlations amongst inputs. Correlated features that appear to be redundant are not removed.
- The square error measures are biased towards the trained network parameters. For example, if the trained neural network cannot represent the system accurately or has low accuracy, it would significantly affect the feature selection algorithm.
- The sensitivity measure of this method is improperly defined. This limitation will be elaborated in Section 6.2.3.

(d) Partial Retraining

Partial retraining [Laar 1997], [Laar 1999] assumes that the neural network being trained with an existing set of input variables has constructed a good representation of the input-output relationship in its hidden layers. The new network is estimated by partially retraining its weight; such that its hidden-layer activities are as close as possible to the hidden-layer activities of the original neural network.

Partial retraining is combined with a modified version of backward elimination for input feature selection. It removes the least relevant variables one at a time, until all the features are removed. The optimal subset of features is determined as the model in which the error does not differ significantly from the original model.

This method outperforms the relevance determination algorithm studied in Laar's papers; and it provides a method in accelerating the retraining process. Some of the possible limitations of this method are identified as follows; namely:

- It is difficult to extend this algorithm to non-neural network methods; i.e., any fuzzy system or decision tree system. This is because this algorithm utilizes the characteristics of the neural network learning algorithm, which differs from other non-neural network methods.

- All the weights are retrained and updated based on the assumption that the original weights give an adequate representation of the input-output relationship of the system. This assumption is generally valid. However, this method has relied too heavily on the original weights, if the original weights have reached a local optimum, it is impossible for the subsequent adjustment to outperform the existing model.
- Although this method speeds up the retraining process, it still poses a computational burden over other methods that does not involve retraining, i.e., sensitivity-based pruning (SBP).

6.1.3 Dimensions of Feature Selection methods

The review of feature selection methods for regressive tasks is non-exhaustive in the previous section. However, it provides some general guidelines for reviewing different aspects of a feature selection algorithm. In this section, dimensions of a feature selection method are contrasted. Some of the dimensions to be considered are derived from [Liu 1998]. A brief survey is undertaken by identifying some possible directions or remarks on the existing techniques.

(a) Application Domain

The two main application domains of feature selection methods are classification and regression tasks. Classification problem has been well studied in the data mining communities. For regressive problem, most of the current emphasis is on the use of neural network as the induction algorithm; feature selection using other induction or inference schemes is still at the nascent stage of research. Regressive feature selection problem is more challenging than the classification task feature selection problem, since no prior knowledge of the system output dynamics is available prior to the learning process. Normally, accuracy is the selection criterion for a regressive feature selection problem.

(b) Criteria of Redundant Features

There are two types of redundant features, namely the irrelevant and the correlated features. Most of the algorithms can effectively remove irrelevant features, but some are ineffective in removing correlated features. There are different degrees of relevancy and correlation measures, some algorithms attempt to rank the features, for example, SBP [Utans 1991], Relief [Kira 1992a]; some are biased towards identifying the strongest relevant features, for example, FOCUS [Almuallim 1991].

(c) Induction Algorithm

There are numerous induction algorithms in machine learning, ranging from classical methods like decision tree and statistical methods to modern methods like Bayesian classifier, connectionist approaches (neural networks), and fuzzy inference systems. In [Blum 1997], four classes of models have been used to define the interaction of the induction algorithm with the feature selection methods; namely: embedded, filter, wrapper, and feature weighting methods. However, feature weighting methods have a different characteristic than the other three classes of feature selection methods. These methods rank the features on the basis of some weight instead of providing an optimal combination of selected features. Therefore, there are three remaining classes of approaches to define the interaction between the feature selection algorithm and the induction algorithm.

The wrapper approach attempts to wrap the induction algorithm with the feature selection method, thus using the induction algorithm purely to derive the performance measures for selection purposes. Most of these methods are tightly coupled with the induction algorithm, i.e., a feature selection method designed for a decision tree induction algorithm cannot be easily substituted with a neural network algorithm. Special care is needed to avoid a tight coupling of the feature selection method with the induction algorithm. Such that the feature selection method is independent and applicable to various induction algorithms.

(d) Selection Criteria

The primary goal of a feature selection problem is to *minimize the number of selected features* to simplify the learning system while *maximizing the predictive accuracy* of the learning system. Therefore, it is reasonable that the predictive accuracy is an important measure for feature evaluation. Most of the criteria for selecting a good feature set have to maximize the predictive accuracy, either directly or indirectly.

Liu and Motoda [Liu 1998] have summarized some selection criteria for a classification system.

- **Information measures:** Information was first proposed as a measurement of the uncertainty of the receiver when it receives all messages. An information measure is referred to as the uncertainty function concerning the true class. The larger value of information measure represents higher levels of uncertainty, and vice-versa. A feature should be selected if it can reduce more uncertainty. A commonly used uncertainty function is the Shannon's entropy.

- Distance measures: This category of measures is also known as separability, divergence or discrimination measures. Algorithms using these measures try to find the features that can separate two classes as far apart as possible.
- Dependence measures: This category of measures is also known as association or correlation measures. It is to quantify how strongly two variables are associated with each other. In classification task, it measures the correlation between each individual input features and output classes.
- Consistency measures: Consistency measures attempt to determine a minimum number of features that separate the classes as consistently as the complete set of features is able to.
- Accuracy measures: Accuracy measure is a good measure for regressive task. Since the goal of the feature selection algorithm is to maximize the predictive accuracy, accuracy measures are directly applicable as a selection criterion of the feature selection. This is the most efficient measure for wrapper model and for regression task problems as it is used for learning of the desired system and the selection of the features.

(e) Search Strategy

Feature selection can be viewed as a search problem, with each state in the search space specifying a subset of the possible features. There are two issues to consider; namely: the search direction and the search method.

Search direction can be in any of the following order: forward, backward, bi-directional or random generation. Forward generation starts with an empty set of features, and sequentially adding new features based on the search method. Backward generation starts with a full set of features and removing one feature at a time. Bi-directional generation takes advantage of the first two approaches by simultaneously starting the search in both directions, and stopping when the best features are identified or the search efforts meet in the middle of both directions of the search space. Random generation searches in a random manner, and is able to avoid falling into local optima.

Three searching methods have been widely adopted; namely: exhaustive/complete search, heuristic search and non-deterministic/instance/random-based search. For an N -feature system, the computational cost for an exhaustive search is $O(2^N)$; and $O(N)$ for heuristic search if the heuristic generation time is ignored. The computational time for non-deterministic search depends on the terminating condition, for an average of k -instances with every N -features, the complexity is $O(kN)$. Another strategy other than the usual search

techniques is the feature weighting technique. These techniques attempt to assign a weight to each feature instead of giving the best subset of features.

(f) Complexity

The computational and space complexities of an algorithm are highly dependent on the search strategy being employed. Random-based methods often have lower computational cost than other methods. In most induction algorithms, because of the non-deterministic learning behaviour, some feature selection algorithms attempt to retrain the system after every feature generation; hence, this accounts for a high computational cost.

Having discussed the different perspectives of a feature selection algorithm, the four feature selection algorithms discussed in Section 6.1.2; namely: the ANFIS selection, Setiono’s feature selection algorithms, Sensitivity-Based Pruning (SBP), and Partial Retraining; are categorized in Table 6-1 based on its corresponding dimensions.

Table 6-1: Dimensions of Feature Selection Algorithms for Regression Tasks

Algorithm	Application Domain		Redundant Data		Induction Algorithm
	Class	Regression	Irrelevant	Correlated	
ANFIS	Yes	Yes	Fixed no. of features		FNN(ANFIS)
Setiono	Yes	Yes	Yes	Yes	NN
SBP	Yes	Yes	Yes	No	NN*
Partial R.	Yes	Yes	Yes	Yes	NN
MCES	Yes	Yes	Yes	Yes	*

Algorithm	Selection Criteria	Search Strategy	Search Method	Need of Retraining	Complexity
ANFIS	Accuracy	Exhaustive	Complete	Yes	High
Setiono	Accuracy	Backward	Heuristic	Yes	High
SBP	Accuracy	Ranking	Random	No	Low
Partial R.	Accuracy	Backward	Heuristic	Yes	Medium
MCES	Accuracy	Ranking	Random	No	Low

SBP has the lowest computational complexity and can be easily decoupled from the induction algorithm (* in the table represents independence from the induction algorithm). The computational complexity of an algorithm is dependant on the search methods, strategy and the need for retraining, as discussed in the previous section. Performance evaluations of some of the algorithms are to be reviewed with benchmarking problems in Section 6.3.

The final row represents the novel feature selection algorithm that will be proposed in this section; namely, the *Monte Carlo Evaluative Selection* (MCES). It has the strengths of low computational complexity and possible decoupling from any induction algorithm. Ranking is

done based on the weighting of different features. The discussion and benchmarking results of this algorithm are highlighted in the subsequent subsection.

6.2 Monte Carlo Evaluative Selection

A computational learning or soft computing type of induction or inference system is generally stochastic and non-deterministic. Therefore, an exhaustive performance evaluation of a partially trained system is redundant during a feature selection process. Instead, the effectiveness of the feature selection algorithm is highly proportional to the number of trials attempted during the different learning states of the system. A trial and error method that depends on training instances may be a superior candidate to an exhaustive evaluation method in this sense.

Most wrapper based feature selection algorithm assumes that the trained system has a good representation of the knowledge of the task. Although this assumption is weak, it is generally acceptable in practice. The goal of the feature selection algorithm is to extract the significance of each feature based on the knowledge inherently present in a system after the training process. Retraining of the system after each search or trial of different sets of input features may avoid being trapped in local minima, and is able to isolate the effects of correlated features. However, the computational efficiency of such algorithms is too costly as compared with those that do not involve a retraining phase. Algorithms such as partial retraining [Laar 1997] attempt to speed up the retraining process. However, they are based on the same assumptions as a random-based method without retraining; hence, suffering from the same limitations (namely, entrapment in local minima, and inclusion of correlated features) as in a random-based method.

On the other hand, classifying the features as irrelevant, weakly relevant and strongly relevant is biased towards some predefined threshold values. Therefore, ranking the features with weights is reasonable as opposed to the identification of a minimal set of relevant features. It is then up to the system designer to choose the number of features to include based on the rank of the features.

A system need not be completely transparent to the feature selection algorithm. Furthermore, the learning system is usually designed to be biased towards the needs of the application. A feature selection algorithm that is tightly coupled with a specific induction algorithm may

pose a problem when a different induction algorithm is adopted for the application. Thus, it is cost effective in terms of resources and time if a feature selection algorithm is easily decoupled from the induction algorithm and is subsequently reformulated to work on other induction algorithm.

On the basis of these above-mentioned rationales, a novel feature weighting algorithm, termed as the *Monte Carlo Evaluative Selection* approach (MCES) is proposed in this subsection.

6.2.1 Assumptions for MCES

The following are the assumptions employed in the design of the MCES approach.

- The induction algorithm is well defined. The trained system is assumed to have a good representation of the underlying knowledge of the task by monitoring its accuracy. When incomplete knowledge is being presented to the system, based on the degree of relevance, the system performance may degrade. This is a general assumption because the feature selection is used to extract the prior knowledge presents in the trained system. In cases where the number of features is too large (about >30), the induction algorithm may be trapped in some suboptimal training state, and thus the feature selection algorithm may suffer from degradation in accuracy. However, under such a situation, feature weighting methods may not be a suitable candidate. Instead, filter approaches or feature extractions that involve human expertise are needed to refine the raw feature set. Subsequently, the MCES approach will accurately refine the feature set to a set of less than ten variables such that some computational learning approach can be applied to learn the system. This workaround is discussed with an application in [Tung 2005].
- The trained induction system has the ability to identify and exclude correlated features. However, in practice, this may not always be possible, most computational learning algorithms (e.g. neural network) will take into account both correlated features in the system if they are presented.
- The method of disabling an input feature is assumed to behave alike removing the actual feature. Sometimes it is impractical to physically remove the input feature during the feature selection process. Therefore, some equivalent method of disabling the input feature without actually removing it is implemented. When an input feature is disabled, it will have minimal influence on the output of the system.

6.2.2 MCES Algorithm

Monte Carlo simulation dates back to the 1940s when physicists at Los Alamos devised games of chance that they could study to help understand complex physical phenomena relating to the atom bomb [Sutton 1998]. Monte Carlo methods are online simulation methods that learn from experience based on randomly generated simulations, without the need for complete knowledge of the environment. Given a random set of experiences (or trials), with guarantee from the *Central Limit Theorem*, the simulation results will eventually converge when each state is encountered for an infinite number of trials.

The methods are briefly described as follows:

Given a random set of features, evaluative feedback is collected when one of the feature states is changing from enabled to disabled, and vice-versa. The evaluative feedback is collected for each instance of example presented to the induction algorithm based on the state of the feature sets. The mean value of evaluative feedback for each feature will eventually converge by averaging these evaluative feedbacks over a number of trials. The converged mean value represents the estimated degree of relevancy of that particular feature over other features. The MCES algorithm is formally listed in Algorithm 6-1.

Algorithm 6-1: Monte Carlo Evaluative Selection Algorithm

- i. Generate a random mask of the feature set, mask 1. (i.e. xxxxxxxx...)
- ii. Select a random position of the feature set.
- iii. Invert a single position of the generated mask, mask 2. (i.e. xxx $\bar{x}$ xxxxx...)
- iv. Get an instance of example.
- v. Example 1 = Apply mask 1 to the example.
- vi. Example 2 = Apply mask 2 to the example.
     For mask with an enabled (1) position, preserve the value of the input feature.
     For mask with a disabled (0) position, perform the Average Substitution.
- vii. Compute output 1 with example 1 using the induction algorithm.
- viii. Compute output 2 with example 2 using the induction algorithm.
- ix. If output 1 is nearer than output 2 to the target output
     Give a positive feedback to mask 1 over mask 2, for that selected position.
   Else
     Give a negative feedback to mask 1 over mask 2, for that selected position.
- x. Repeat steps (i) to (ix) for all examples for an empirically decided number of iterations.
- xi. Sum and average the positive and negative feedbacks for each individual position.

Mathematically, the algorithm is described in equations (6-4)-(6-9). The notations in this section are independent of those defined in Sections 2 - 5 and are only applicable to this section. Some of the notations are borrowed from the SBP formulations discussed in Section 6.1.2.

Assume that there are J input features, with I training instances, and,

- $\mathbf{X}$ represents the matrix of the input variables,
- $\mathbf{x}_j$ represents the transpose vector of all exemplars for feature j , where $\mathbf{x}_j = [x_{1j} \ x_{2j} \ \dots \ x_{ij} \ \dots \ x_{Ij}]^T$,
- $\overline{x_j}$ represents the average value of all elements in $\mathbf{x}_j$,
- $\mathbf{x}_i$ represents the vector of all input features for exemplar i , where $\mathbf{x}_i = [x_{i1} \ x_{i2} \ \dots \ x_{ij} \ \dots \ x_{iJ}]$,
- x_{ij} represents the element of feature j at exemplar i , $x_{ij} \in \mathbf{X}$,
- $\mathbf{m}_i$ represents the mask vector for exemplar i , where $\mathbf{m}_i = [m_{i1} \ m_{i2} \ \dots \ m_{ij} \ \dots \ m_{iJ}]$,
- m_{ij} represents the mask of feature j at exemplar i , $m_{ij} \in \{0, 1\}$,
- R_j represents the average feedback for feature j ,
- r_{ij} represents the positive/negative feedback of x_{ij} ,
- W represents the tunable parameter set of the induction algorithm, and
- $E(.)$ represents the error function (difference between the predicted and actual output).

As described in steps (i)-(ix) of Algorithm 6-1, the feedback is computed using equation (6-4).

$$r_{ij} = \begin{cases} \overbrace{E(\overline{x_j}, \text{Mask}(\mathbf{x}_i, \mathbf{m}_i), W) - E(x_{ij}, \text{Mask}(\mathbf{x}_i, \mathbf{m}_i), W)}^{\text{positive feedback}}, & \text{if } |E(\overline{x_j}, \dots)| \geq |E(x_{ij}, \dots)| \\ \underbrace{-[E(\overline{x_j}, \text{Mask}(\mathbf{x}_i, \mathbf{m}_i), W) - E(x_{ij}, \text{Mask}(\mathbf{x}_i, \mathbf{m}_i), W)]}_{\text{negative feedback}}, & \text{if } |E(\overline{x_j}, \dots)| < |E(x_{ij}, \dots)| \end{cases} \quad (6-4)$$

The masking operation is performed with equation (6-5).

$$\text{Mask}(\mathbf{x}_i, \mathbf{m}_i) = \text{Mask}([x_{ij}], [m_{ij}]), \text{ and } \text{Mask}(x_{ij}, m_{ij}) = \begin{cases} x_{ij}, & \text{if } m_{ij} = 1 \\ \overline{x_j}, & \text{if } m_{ij} = 0 \end{cases} \quad (6-5)$$

and the *Average Substitution* is performed as shown in equation (6-6).

$$\overline{x_j} = \frac{1}{I} \sum_{i=1}^I x_{ij} \quad (6-6)$$

The usage of masking in feature selection to temporarily change the feature states is common in the literature. Naturally, the masking operation in equation (6-5) is proposed to disable the effect of certain features by substituting it with the average value computed in equation (6-6). Some other masking operations are similarly proposed in GA-based RBF networks [Fu 2002a], [Fu 2002b]. Although the computations are different due to the algorithmic differences, the motivation is similar in disabling or enabling the features with respect to the state of the mask.

The relevancy of feature j is determined by the average feedback, or the expected value of r_{ij} . The average feedback of feature j is computed with equation (6-7).

$$\begin{aligned} R_j &= E[r_{ij}] \\ &= r_{ij} P(r_{ij}) \\ &= \sum_{\mathbf{x}_i, \mathbf{m}_i, W} r_{ij} P(r_{ij} | \mathbf{x}_i, \mathbf{m}_i, W) P(\mathbf{x}_i, \mathbf{m}_i, W) \end{aligned} \quad (6-7)$$

Assuming each condition has equal probability, then for a given tunable parameter set of induction algorithm W , the average feedback is computed with equation (6-8).

$$R_j = \frac{1}{I} \sum_{i=1}^I r_{ij} \quad (6-8)$$

The average feedback or weight of each feature represents the relative importance of that feature over the other features. Negative or near-zero weight indicates that the corresponding feature is irrelevant. Higher weight value means that the particular feature is of higher importance.

If the error function is taken as the difference between the actual and the predicted outputs, then equation (6-4) is simplified to equation (6-9).

$$\begin{aligned} r_{ij} &= \begin{cases} |(y_1 - y) - (y_2 - y)|, & \text{if } |E(\overline{x_j}, \dots)| \geq |E(x_{ij}, \dots)| \\ -|(y_1 - y) - (y_2 - y)|, & \text{if } |E(\overline{x_j}, \dots)| < |E(x_{ij}, \dots)| \end{cases} \\ &= \begin{cases} |y_1 - y_2|, & \text{if } |E(\overline{x_j}, \dots)| \geq |E(x_{ij}, \dots)| \\ -|y_1 - y_2|, & \text{if } |E(\overline{x_j}, \dots)| < |E(x_{ij}, \dots)| \end{cases} \end{aligned} \quad (6-9)$$

where,

y represents the actual output,

y_1 represents the predicted output with the 1st mask being applied to the input, and

y_2 represents the predicted output with the 2nd mask being applied to the input.

It is justifiable to adopt this error function rather than the usual *squared error* function in this feature selection. The rationale will be discussed in the Section 6.2.3.

One of the assumptions is that, the underlying induction algorithm is well defined, as discussed in Section 6.2.1. However, this may not be generally possible. Furthermore, equation (6-8) is conditioned upon the trained tunable parameters of an induction algorithm, which is biased towards some state of the learning process. In order to eliminate this assumption, multiple induction algorithms are needed to derive an unbiased estimation of the resultant feedback values. This is presented in Algorithm 6-2.

Algorithm 6-2: MCES with multiple induction algorithms

- i. *Trained multiple induction algorithms to get different set of trained parameters.*
- ii. *For an epoch:*
- iii. *Choose an induction algorithm.*
- iv. *Apply MCES (Algorithm 6-1).*
- v. *Repeat.*
- vi. *Compute the average of the feedbacks.*

If there are K independent induction algorithms, then equation (6-8) is reformulated into equation (6-10).

$$\begin{aligned}
 R_j &= \sum_{W_k} \frac{1}{I} \sum_{i=1}^I r_{ij} P(r_{ij}, W_k) \\
 &= \sum_{W_k} \frac{1}{I} \sum_{i=1}^I r_{ij} P(r_{ij} | W_k) P(W_k) \\
 &= \frac{1}{KI} \sum_{k=1}^K \sum_{i=1}^I r_{ij}
 \end{aligned} \tag{6-10}$$

where W_k represents the k^{th} set of tunable parameters of the induction algorithms.

Equation (6-10) demonstrates that the MCES algorithm is independent of the conditional probability of the prediction model's parameters. This is because the effect of the prediction parameters is being averaged over the learning trials. This algorithm provides a useful estimate of the average feedback measure, and is independent of any underlying induction algorithm used to perform the prediction for a particular task. The MCES algorithm can be

readily applied to any task as long as some form of accuracy or performance indication can be defined and derived from the learning system.

6.2.3 Comparison with Sensitivity-based Pruning

There are two main differences between the MCES and SBP methods. Both methods attempt to find the sensitivity effect between enabling and disabling of an input feature, with the effect being measured by means of the accuracy measures.

(a) Error function

Consider a scenario where the three types of output appeared in a distance space as shown in Figure 6-1. The notations are defined as follows; namely:

- y represents the actual output,
- y_1 represents the predicted output with the 1st mask being applied to the input,
- y_2 represents the predicted output with the 2nd mask being applied to the input,
- y_2' represents another predicted output with the 2nd mask being applied to the input,
- e_1 is the error function between y_1 and y ,
- e_2 is the error function between y_2 and y , and
- e_2' is the error function between y_2' and y .

Figure 6-1: Graphical representation of error function.

Both MCES and SBP methods attempt to measure the differences between the outputs when one of the features is changing its state, from enabled to disabled or vice-versa. Consider y as the actual output, y_1 is the output when one of the features is disabled, and y_2 (or y_2') is the predicted output when the same feature is enabled. By applying the sensitivity measure of SBP as in equation (6-1), the result is indifferent between the two predicted outputs (y_2 and y_2'), since the squared of both errors (e_2 and e_2') are identical. However, y_2' yields a higher relative sensitivity change as compared with y_2 , since its distance is further away from y_1 . MCES considers this by computing the feedback as a direct distance between the two predicted outputs as described in equations (6-4) and (6-9).

(b) Fixed mask vs. Random mask

Figure 6-2: Masking of SBP and MCES

As shown in Figure 6-2, SBP uses a fixed mask with all features being enabled, and measuring the effect when disabling a particular feature. If a high correlation exists between the disabled feature and the other features, then the disabled feature is classified to be redundant because of the very low sensitivity measure, even though it is not. Therefore, the measurement of a feature weight in SBP is conditioned and biased towards the other features.

However, in the MCES approach, all the other feature masks are randomized, so effectively the other correlated features may or may not affect the computation of this feedback. This minimizes the dependency and bias of a particular feature towards the other features. If there are n highly correlated features with the same degree of relevance R , and then the relevance measures will be evenly distributed across the n -features by using MCES. The actual relevance being measured is R/n for each of the correlated features. Hence, the MCES method is able to produce a proper assessment of the correlation within the set of input features.

6.3 Benchmarking Results

The MCES algorithm is simulated for both the classification and the regression tasks in this subsection. There are two induction algorithms being adopted in these benchmark studies; namely: the multi-layer-perceptron (MLP) with momentum backpropagation learning rule, and the ANFIS model [Jang 1993]. However, this can readily be extended to include other induction algorithms. All the MLP models in the benchmark studies have a three-layer structure, with the number of hidden layer nodes being doubled that of input nodes. All the ANFIS are modelled in Matlab with the subtractive clustering option present in Matlab.

6.3.1 Rule-plus-exception problem

The first dataset is a six-dimensional version of the rule-plus-exception problem [Mozer 1989], [Laar 1999]. The relevancies of the six binary input variables, $X_1, X_2, \dots, X_6$, of this classification problem are designed to be easily distinguishable. The binary output is given by equation (6-11).

$$Output = X_1X_2 + \overline{X_1X_2X_3X_4} \tag{6-11}$$

As inferred from equation (6-11), X_1 and X_2 are strongly relevant to the target output; X_3 and X_4 are weakly relevant; X_5 and X_6 are not relevant to the output. There are 64 possible training patterns. The induction algorithm used in this benchmark is an MLP network, which has been trained for 1000 iterations with all the six inputs prior to applying the MCES algorithm for 10 training epochs. The average weights of each of the six input features derived from three independent runs of MCES algorithm are plotted in Figure 6-3.

Figure 6-3: Average weights of the six features for the RPE problem.

For every run, an MLP is trained prior to the MCES feature selection process. It is shown in Figure 6-3 that the weights of relevant features differ significantly from those of irrelevant features. This correlates with the output expressed in equation (6-11).

6.3.2 Automobile MPG Prediction

This second experiment describes the application of MCES in a nonlinear regression problem. In the automobile MPG (miles per gallon) prediction, an automobile’s fuel consumption in terms of MPG is predicted by the ANFIS model based on six different

characteristics. The same benchmarking was done with the ANFIS input selection method mentioned in Jang’s paper [Jang 1996]. The six features are listed in Table 6-2.

Table 6-2: Six features of the automobile MPG prediction problem.

Features	No. of cylinders	Displacement	Horsepower	Weight	Acceleration	Model year
Value	discrete	continuous	continuous	continuous	continuous	discrete

The output to be predicted with respect to the six features is the fuel consumption in MPG. The best two features derived from Jang’s ANFIS input selection method are the “weight” and “model year” features [Jang 1996].

In the benchmarking of MCES, the ANFIS model was trained once prior to the MCES selection process. The ANFIS was trained with the subtractive clustering option (cluster width of 0.2) and was trained for 10 epochs. The selection process of MCES took 10 iterations for each trial. The 392 instances of data were randomly divided into training and test sets of equal size. Six trials were conducted, three being on the training set, and three on the testing set.

Figure 6-4: MCES algorithm with ANFIS model on the MPG prediction problem.

Figure 6-4 plots the average weight over the six trials. “Displacement” has been identified as the most important feature, and subsequently followed by the “model year” and the “acceleration”. The result differs from those reported in Jang’s paper. A validation was performed to demonstrate the validity of the result. This is shown in Figure 6-5.

Figure 6-5: Simulation of ANFIS with different input feature set.

As shown in Figure 6-5, two ANFIS simulations with subtractive clustering were performed with different sets of the input features. The first set is the best feature set reported by the ANFIS feature selection method [Jang 1996], which is the “weight” and “model year” feature set. The second set is the best feature set derived from MCES selection, which is the “displacement” and the “model year” feature set. Each set of features was trained for five epochs with ANFIS, and was subsequently validated once with the testing data. The root mean-squared errors (RMSE) of the simulation were recorded. This forms the Y-axis of the chart in Figure 6-5. The X-axis represents the training epoch.

The RMSE for features selected by Jang’s method are 130.06% higher than the features selected by MCES for the test dataset. This shows that the MCES algorithm performs superior to the ANFIS input selection method. Moreover, the ANFIS input selection method is exhaustive, which is much slower than the random-based MCES method. The training RMSE for both feature sets are similar in the first two training epochs. Thus, the ANFIS input selection is prone to incorrectly selecting the non-optimal feature set based on the RMSE after the first epoch. However, in the consecutive training epochs, the second feature set outperformed the first feature set. This violates the assumption described by ANFIS input selection method, in that the ANFIS model with the smallest error after one epoch of training has a greater potential of achieving a lower error when more epoch of training are involved. This is causing the performance deficiency in this experiment.

This benchmarking study demonstrates that the MCES algorithm is more robust and is computationally more effective than the ANFIS input selection method for the automobile MPG prediction problem. Both methods are similarly applicable to nonlinear regressive tasks.

6.3.3 Audio Detection problem

The audio detection problem is a classification problem; there are audio samples that represent different categories of audio signals. Ten features are extracted from the audio samples, and the goal of the task is to classify the data into different audio categories. Without loss of generality, only two audio categories are considered in this problem; that is: the “Fire Alarm” and the “Glass Breaking” classes.

After some pre-processing steps, the features are extracted and normalized, these ten extracted features are the zero-crossing rate (ZCR), average magnitude difference function (AMDF), average energy (AveEnergy), variance (Var), and five features from the results of the fast Fourier transform of the signal; namely: peak1, peak2, peak3, centroid, std1, and std2. The details of the problem can be found in Ting’s report [Ting 2001]. The goal of feature selection is to select or rank the best features, to reduce the size of input features appropriately.

There were 1612 examples for training. The underlying induction model is the MLP, which was trained for 1000 iterations prior to the MCES selection process, and each trial consisted of two MCES learning iterations.

Figure 6-6: Average weight and ranking of MCES on the audio detection problem.

Three independent trials were performed on this task. The average weights are plotted in Figure 6-6. Figure 6-6 shows the ranking of the most relevant features. The most relevant feature is the ZCR, and then follows by the AMDF, the Centriod and the Peak1.

Table 6-3: Features selected with different feature selection algorithms.

Features	ZCR	AMDF	AvgE	Var	Peak1	Peak2	Peak3	Centro.	Std1	Std2
FOCUS	0	1	1	0	0	0	0	0	0	0
Relief	-0.2	0	0.1	0.1	0.8	0.1	0.2	0	0.2	-0.1
Ting's result	1	0	1	0	0	0	0	1	1	0

Table 6-3 lists the feature selection results on the same dataset. The first and the second rows list two popular filter model feature selection results; namely: the FOCUS [Almuallim 1991] [Almuallim 1994] and the Relief [Kira 1992a] [Kira 1992b] methods. FOCUS is a min-bias algorithm; it attempts to derive the minimum set of features for correct classification. Relief is a feature ranking method; it aims to rank the features. The third row presents one possible set of features reported in Ting's report [Ting 2001]; the features were manually selected based on the accuracy measures of some of the possible combinations of these feature sets.

Figure 6-7: Performance comparison of the feature selection algorithms.

Figure 6-7 lists the performance comparison of these different feature selection algorithms. They are quantified with the mean-squared-errors (MSE) and the classification rates. These combinations of features are selected based on the results from Figure 6-6 and Table 6-3. These selected features were trained with MLPs with momentum learning rate for 1000 epochs. Seven combinations of features are being presented; the number in bracket (.) represents the number of features being used in that particular combination. The feature set selected for the Relief algorithm is the set of three highest-ranking features in Table 6-3. There are three different combinations for the features selected by MCES; which are the two-best, the three-best, and the four-best features with respect to the ranking presented in Figure 6-6. As shown in Figure 6-7, the combination of the two-best features selected with MCES

outperformed those selected using the FOCUS and the Relief algorithms. The MCES selected features with three and four best features can achieve a similar performance as Ting's manual selection method; and are able to achieve a similar performance to the one with all the ten input features. Based on the classification rates and MSE in this experiment, the best set of features is the three highest ranking features selected by the MCES algorithm.

6.3.4 Stock market prediction

The final problem is a prediction task of future stock market condition based on current situations. It is well known that a stock price is a chaotic signal. Hence, it may not be easily predicted even with the existence of expert knowledge. The feature selection is performed on this task where the output signal is chaotic in nature, or in the presence of non-deterministic, nonlinear regressive signal.

The dataset for this problem is taken from one of the trading counters of Shanghai stock exchange. It represents the everyday trading condition since the year 1992. The raw dataset has a total of six features; namely: the date, the opening price, the highest and the lowest price in a single trading day, the closing price, and the total number of shares traded (volume) in that trading day. The data are pre-processed into a ten-feature vector; namely: the normalized opening, highest, lowest, closing prices and trading volume, the other five elements are the recent trend information of these first five normalized elements. This trend information is included such that the data can be presented without being in the original time sequence. The objective of the induction algorithm is to predict the next day's highest price. This set of features is weakly-defined. As shall be demonstrated subsequently, the MCES algorithm is able to perform equally well on this problem with the available ten weakly-defined features. There are two significances in this experiment; firstly, the underlying induction algorithm is weakly trained, since the features are incomplete for this prediction task. Secondly, the input features are highly correlated.

In the benchmark simulation, two induction algorithms were wrapped with the MCES selection approach, namely the MLP with momentum learning and the ANFIS model. Both models were trained prior to the MCES selection process. The MCES algorithm learnt for two iterations to arrive at the resultant weights.

Figure 6-8: Benchmarking results for the stock prediction problem with MCES.

Figure 6-8 shows the average weight of the MCES algorithm for three independent trials. It is known that the first four features are highly correlated, nevertheless, the MCES algorithm is able to decorrelate the features and to rank the two most important out of the ten input features. The two selected features are the normalized closing and highest prices.

Figure 6-9: Performance comparison with different combinations of features.

Figure 6-9 shows the performance in terms of mean squared errors of the learning models with different combinations of input features; namely: the first with all the ten features, and the second with the two features selected by MCES. For MLP learning, the learning mean-squared-error was 0.013207 with all the ten input features. In contrast, the learning with the only two input features produced an average MSE of 0.013554. These two results are highly

similar, with insignificant difference of 2.63% in accuracy. For ANFIS learning, the result shows that the set of two features was able to perform comparatively well with only 6.11% decrement in performance. This decrease in accuracy is insignificant. Furthermore, the learnt ANFIS model with ten input features is impossible to be comprehended with a significant degradation in expressiveness. However, the ANFIS model with the only two input features has better model interpretability. These improve the expressiveness and interpretability of the ANFIS model, which is a significant design criterion of the fuzzy model as described in Section 2.2.2. This implies that MCES can be used as a general tool in reducing the set of redundant features prior to the learning process with any TSK models. Having the reduced set of features, the interpretability of the TSK model can be substantially improved. As the FITSK models proposed in this Thesis and ANFIS both belongs to the class of TSK models, MCES is similarly effective in improving the interpretability of these models.

This benchmarking result conclusively demonstrates that the MCES algorithm can accurately extract the two best features out of a weakly-defined combination of ten features, which consists of correlated and irrelevant features.

6.4 Summary Remarks

This section reviews the feature selection problem from the two application contexts; namely: on the classification and regressive tasks. In the area of the regressive task feature selection problem, the research has been very limited and mostly emphasizing on the pruning algorithm of the underlying model. This section proposes a novel wrapper-based feature selection method, the Monte Carlo Evaluative Selection method. The method works by summarizing the feedback of errors generated by selecting different sets of features. MCES possesses the strengths of the following; namely:

- easy decoupling from the underlying induction algorithm;
- applicability to both classification and regressive tasks;
- ability to identify both irrelevant and correlated features;
- high accuracy in general tasks due to its minimal reliance on relevancy measure;
- improving the model interpretability of the system; for example, the TSK model.

These strengths have been highlighted with four benchmarking problems; namely: the rule-plus-exception problem, the automobile MPG prediction, the audio detection problem, and the stock market prediction. In classification task such as the rule-plus-exception problem and

the audio detection problem, the MCES method has been demonstrated to have better results over two classical feature selection methods and a manual selection method. In regressive task such as automobile MPG prediction, the MCES method outperformed the ANFIS feature selection method by 130.06% in terms of accuracy in the validation of test dataset. Finally, the benchmark of stock market prediction has demonstrated that the MCES method is able to correctly identify correlated features. Furthermore, the results have shown that the MCES method has selected the two best features among the ten original features. This is made possible with insignificant prediction accuracy degradation of about 2.63% to 6.11% mean-squared-error. This is comprehended with a significant improvement on the expressiveness of the ANFIS model with this minimally selected feature set. Apart from these four benchmarking problems, the MCES algorithm has been successfully applied to a number of practical applications. These are reported in [Tung 2003], [Tung 2005] for biomedical applications, [Jiang 2003] for bank failure prediction, and [Tan 2003] for ovarian cancer classification.

MCES is designed as a wrapper-based feature weighting method that ranks the features accordingly instead of providing an optimal subset of features. However, it is possible to implement a judicious selection of an optimal subset of features. Some insights are established in Fu's paper [Fu 2003], where bottom-up and top-down searches are performed to decide on the practical subset of features. The classification error rates are monitored during the addition or removal of features, the feature subset with minimal error is subsequently selected in [Fu 2003]. It is possible to induce similar error curves from the results of MCES. The behaviour of the error curves during the addition or removal of features will be used to acquire information on selecting the optimal feature subset. This would require some further efforts in characterizing and experimenting with the benchmark problems in this section.

7 Conclusions and Recommendations

*"Hoary fishers and woodcutters, along the banks, find calm water, in autumn moon or in spring wind.
By the wine jars, fill porcelain: Discuss talk and tale, only laugh and gale..."*
~ Luo Guanzhong

The conclusions and recommendations in this section describe a high-level view of topics presented in the Thesis. For details on the specific topics, please refer to the summary remarks in Sections 3 - 6.

7.1 Conclusions

In this section, a blended view of research issues in designing fuzzy reinforcement learning models is addressed. The research issues are distributed across three areas in machine learning research. These areas are as follows; namely: fuzzy modelling; reinforcement learning; and feature selection.

Figure 7-1: A blended view of research areas in this Thesis.

Table 7-1: A summarized view of research achievements in this Thesis.

Area	Models / Algorithms	Addressing Problems	In Sections
(a)	FITSK	Model interpretability, global accuracy, curse-of-dimensionality.	Section 3
	Localized Kalman filter	Parameter learning with superior interpretability, immunity to curse-of-dimensionality.	Section 3.1.3
(b)	Generic RL Framework	Modularity, generality for RL models	Section 4.1
	Maximum Reward RL	Non-cumulative RL tasks.	Section 5
(c)	FITSK-RL	RL in continuous domains, systematically defined value function approximation.	Section 4
(d,e)	MCES	Curse-of-dimensionality and model expressiveness.	Section 6

Fuzzy modelling and reinforcement learning are the main focus of this Thesis, and feature selection serves to complement the design issue of the fuzzy modelling by pre-processing redundant features to achieve greater model interpretability. This is illustrated with the Venn diagram in Figure 7-1, and is tabulated in Table 7-1.

In area (a) of Figure 7-1 and Table 7-1, the design issues of fuzzy systems for the online learning of nonlinear system estimation are tackled. The proposed FITSK framework described in Section 3 has the flexibility of adopting different formulations and settings based on different design needs. The three design criteria are as follows; namely: the model interpretability; the global accuracy; and the immunity to curse-of-dimensionality. Please see also Figure 3-6 for details. The framework is subsequently mapped to a zero-order FITSK model and a first-order FITSK model. The zero-order FITSK model is suitable for fast adaptive system without a major consideration of model interpretability. The first-order TSK model with global Kalman filter is appropriate for low dimensionality, low interpretability systems with optimal global learning accuracy. A localized Kalman filter algorithm has been proposed and proven to achieve optimal accuracy for locally weighted least-squares error criterion. It has also been demonstrated to have superior immunity to the issue of curse-of-dimensionality. A guideline is subsequently formulated based on the three design criteria for model selection of these FITSK models. Five benchmarking studies were performed to demonstrate the design feasibilities of the FITSK framework with respect to these design criteria.

In area (b) of Figure 7-1 and Table 7-1, two possible contributions in reinforcement learning research are achieved. These two possible contributions are as follows; namely: the proposed generic reinforcement learning framework; and the maximum reward criterion for the learning of non-cumulative reward reinforcement learning. The generic reinforcement learning framework employs a modular design with three modules; namely: the function approximator; the policy generator; and the TD generator. The key strengths of the generic reinforcement learning framework are its modularity and generality. The modularity provides functional independencies among the modules, such that reformulating a single module would not affect the operations of the other modules. The generality applies to the generalization of different popular reinforcement learning algorithms, whereby TD learning, SARSA learning and Q-learning can be readily mapped into such a framework with ease.

Furthermore, the proposed maximum reward reinforcement learning is a possible new learning paradigm that exhibits superior performance for problems with underlying non-cumulative nature of reward values. Subsequently, the maximum reward reinforcement learning is mapped to the generic reinforcement learning framework to solve the modelling issue of this new learning paradigm. The distinct characteristic of the maximum reward reinforcement learning is a non-cumulative, maximum reward-oriented behaviour of the learning agent, as opposed to the usual cumulative rewards-oriented behaviour. It attempts to search for the policy that leads to the largest reward, irregardless of intermediate lower rewards. This positions it to exhibit superior immunity in the presence of sub-optimal policies; and is suitable in modelling the asymptotic maximization behaviour of an optimal pricing problem.

Area (c) of Figure 7-1 and Table 7-1 provides an intersection of research efforts for the fuzzy modelling in the reinforcement learning system. The proposed fuzzy reinforcement learning model is termed the FITSK-RL model. FITSK-RL incorporates a zero-order FITSK model into the function approximator module of the generic framework to inherit its modularity and generality. Two motivations drive the adoption of the zero-order FITSK model as the function approximator of the generic framework. Firstly, the problem domain is extended to continuous state-space. Secondly, the fuzzy operations of the FITSK model are defined by the TSK fuzzy inference scheme. This ensures that the operations of the function approximator is systematically defined, and has strong theoretical foundation from the fuzzy inference scheme perspective. The FITSK-RL model is applicable to general classes of temporal learning problems, such as the discrete state-action, continuous state-discrete action, and continuous state-action domains. This is demonstrated using two benchmarking simulations and their result analysis.

Subsequently, the maximum reward reinforcement learning is integrated into the FITSK-RL model by mapping it into the generic reinforcement learning framework. A successful application of the maximum reward FITSK-RL model to asymptotically approximate the supremum of payoff values in an optimal stopping problem is discussed. This problem demonstrates two strengths of the model; namely: effective in modelling non-cumulative reward problem; relaxation of the modelling complexity and thus the improvement of the result interpretability.

In area (d) of Figure 7-1 and Table 7-1, a feature selection problem is studied to complement the curse-of-dimensionality issue prevalent in fuzzy modelling. The proposed MCES approach has been demonstrated to be effective in two application contexts; namely: on the classification and regressive (nonlinear system estimation) task contexts. MCES relies on minimal assumptions, thus it possesses the strengths of the following; namely: easy decoupling from the underlying induction algorithm; ability to identify both irrelevant and correlated features; and low computational complexity due to its random-based learning process. In area (e) of Figure 7-1 and Table 7-1, the MCES approach has been demonstrated to be effective in selecting features for a popular fuzzy model, the ANFIS system. The reduction of features is crucial for improving the interpretability of a fuzzy model, and relaxation from the curse-of-dimensionality limitation.

7.2 Recommendations for Future Research

In areas (b) and (c) of Figure 7-1, the convergence and stability of the maximum reward reinforcement learning and the FITSK-RL model have not been properly established. The current convergence studies are directly extended from the results by Tsitsiklis and Van Roy [Tsitsiklis 1997], [Tsitsiklis 1999], [Van Roy 1998]. This is possible by relating the zero-order FITSK model as a special case of the linear local basis function approximator. However, these convergence results have not been formally derived from the maximum reward reinforcement learning and the FITSK-RL model contexts. A possible direction is to establish the convergence based on theories from Markov decision processes [Puterman 1994], nonlinear dynamics systems [Strogatz 1994], and Hilbert space approximation [Van Roy 1998]. This is significant in ensuring the stability and convergence of these models from the control theory perspective, and thus warranting their successful applications to a wider range of control tasks.

The function approximation of the generic reinforcement learning model is modelled with the zero-order FITSK model. However, the feasibility of using the first-order FITSK model for value function approximation has not been studied in this Thesis. To complicate the problem even further, the first-order FITSK model may possibly suffers from divergence issue in this application context. Van Roy [Van Roy 1998] has pointed out that the divergence issue persisted in the case of nonlinear parameterizations, even though in the presence of some regularity conditions. This generally leaves out the choice of using any higher order function approximator for value function approximation. However, if the regularity conditions can be

discovered, the class of least-squares temporal difference methods (LSTD) [Bradtke 1996], [Boyan 1999], [Boyan 2002], [Xu 2002] can possibly be extended to the value function approximation of the first-order FITSK model.

Furthermore, the discussion is incomplete without mentioning of the future prospect in areas (f) and (g) of Figure 7-1. These areas are best described with applications that need to be accommodated by the mixture of these research efforts. The application of optimal ventilator control in intensive-care-unit (ICU) [Rutledge 1993], [Wang 1998], [Rees 1999] is a proper candidate in this aspect. The foundation work of using reinforcement learning for this application has been achieved by Ding [Ding 2003]. The preliminary proposal for extending this with techniques from this Thesis has been described in [Quah 2004d]. This is a real-life control application with a huge number of attributes, the state-action spaces are continuous, and the outcome is delayed. Thus, the MCES algorithm is needed to reduce the number of attributes; the state-action spaces can be modelled with the FITSK model (area (g)) or some other possible approximation methods (area (f)). It is subsequently modelled as a reinforcement learning task, and is solved using the FITSK-RL model. This proposal for further research [Quah 2004d] is currently an on-going research topic undertaken in the Centre for Computational Intelligence.

Other than these areas, some specific topics of possible future research are highlighted in Section 3.3. This includes formalizing the momentum learning formula of zero-order FITSK model, and applying this formula on an extensive set of benchmark studies. Furthermore, it is possible to simulate this model for the learning of Continuous Variable Transmission (CVT) control and to contrast against Ang's MCMAC [Ang 2000] in practical application.

References

- [Abonyi 2000] J. Abonyi, R. Babuska (2000), "Local and global identification and interpretation of parameters in Takagi-Sugeno fuzzy models", *FUZZ-IEEE 2000*, 2:835-840.
- [Albus 1975] J.S. Albus (1975), "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)", *Journal of Dynamic Systems, Measurement and Control*, 97(3): 220-227.
- [Almuallim 1991] H. Almuallim, T.G. Dietterich (1991), "Learning with many irrelevant features", in *Ninth National Conference on Artificial Intelligence*, MIT Press, 547-552.
- [Almuallim 1994] H. Almuallim, T.G. Dietterich (1994), "Learning Boolean concepts in the presence of many irrelevant features", *Artificial Intelligence*, 69:276-306.
- [Ang 2000] K.K. Ang, C. Quek (2000), "Improved MCMAC with Momentum, Neighborhood, and Averaged Trapezoidal Output", *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 30(3):491-500.
- [Atkeson 1995] C. Atkeson (1995), "Using Local Trajectory Optimizers to Speed Up Global Optimization in Dynamic Programmin", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Baird 1993a] L.C. Baird (1993), "Advantage updating", *Technical Report WL-TR-93-1146*, Wright-Patterson Air Force Base.
http://www.leemon.com/papers/adv_upd/adv_upd.pdf
- [Baird 1993b] L.C. Baird, A.H. Klopff (1993), "Reinforcement Learning with High-Dimensional, Continuous Actions", *Technical Report WL-TR-93-1147*, Wright-Patterson Air Force Base.
<http://www.leemon.com/papers/wirefit/wirefit.pdf>
- [Baird 1994] L.C. Baird (1994), "Reinforcement Learning in Continuous Time: Advantage Updating", *Proceedings of the International Conference on Neural Networks*, Orlando.
- [Baird 1995] L.C. Baird (1995), "Residual Algorithms: Reinforcement Learning with Function Approximation", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Baird 1999a] L.C. Baird, A.W. Moore (1999), "Gradient descent for general reinforcement learning", *Advances in Neural Information Processing Systems*, 11.
- [Baird 1999b] L.C. Baird (1999), *Reinforcement Learning Through Gradient Descent*, PhD thesis, Carnegie Mellon University, Pittsburgh, PA.
- [Bellman 1957] R.E. Bellman (1957), *Dynamic Programming*, Princeton University Press.

- [Berenji 1992] H.R. Berenji, P. Khedkar (1992), "Learning and tuning fuzzy logic controllers through reinforcements", *IEEE Transactions On Neural Networks*, 3(5).
- [Berenji 1994] H.R. Berenji (1994), "Fuzzy Q-learning: a new approach for fuzzy dynamic programming", *Proceedings of the Third IEEE Conference on Fuzzy Systems*, 1:486-491.
- [Berenji 1996] H.R. Berenji (1996), "Fuzzy Q-learning for generalization of reinforcement learning", *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, 3:2208-2214.
- [Berenji 1998] H.R. Berenji, S.K. Saraf (1998), "Competition and collaboration among fuzzy reinforcement learning agents", *The 1998 IEEE International Conference on Fuzzy Systems*, 1:622-627.
- [Berenji 1999] H.R. Berenji, D. Vengerov (1999), "Cooperation and coordination between fuzzy reinforcement learning agents in continuous state partially observable Markov decision processes", *FUZZ-IEEE '99*, 2:621-627.
- [Berenji 2000] H. R. Berenji, D. Vengerov (2000), "Advantages of cooperation between reinforcement learning agents in difficult stochastic problems", *FUZZ-IEEE 2000*, 2:871-876.
- [Berenji 2001] H.R. Berenji, D.A. Vengerov (2001), "On Convergence of Fuzzy Reinforcement Learning", *FUZZ-IEEE '01*.
- [Bertsekas 1996] D.P. Bertsekas, J.N. Tsitsiklis (1996), *Neuro-Dynamic Programming*, Athena Scientific.
- [Bezdek 1981] J.C. Bezdek (1981), *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York.
- [Blum 1997] A.L. Blum, P. Langley (1997), "Selection of relevant features and examples in machine learning", *Artificial Intelligence*, 97:245-271.
- [Bonarini 1996] A. Bonarini (1996), "Delayed Reinforcement, Fuzzy Q-Learning and Fuzzy Logic Controllers", In Herrera, F., Verdegay, J. L. (Eds.), *Genetic Algorithms and Soft Computing, (Studies in Fuzziness, 8)*, 447-466.
- [Boyan 1995a] J.A. Boyan, A.W. Moore (1995), "Generalization in reinforcement learning: Safely approximating the value function", *NIPS-7*. San Mateo, Morgan Kaufmann.
- [Boyan 1995b] J.A. Boyan, A.W. Moore (1995), "Robust Value Function Approximation by Working Backwards", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Boyan 1999] J.A. Boyan (1999), "Least-Squares Temporal Difference Learning", *ICML 1999*.

- [Boyan 2002] J.A. Boyan (2002), "Technical Update: Least-Squares Temporal Difference Learning", *Machine Learning*, 49:233-246.
- [Bradtke 1996] S.J. Bradtke, A.G. Barto (1996), "Linear Least-Squares Algorithms for Temporal Difference Learning", *Machine Learning*, 22:33-57.
- [Breiman 1984] L. Breiman, J.H. Friedmanm, R.A. Olsehn, C.J. Stone (1984), *Classification and Regression Trees*, Wadsworth International Group.
- [Brent 2002] R.P. Brent (2002), *Algorithms for minimization without derivatives*, Dover Publications.
- [Cantu-Paz 2003] E. Cantu-Paz, C. Kamath (2003), "Evolving neural networks to identify bent-double galaxies in the FIRST survey", *Neural Networks*, 16:507-517.
- [Cardie 1993] C. Cardie (1993), "Using decision trees to improve case-based learning", in *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, Inc., 25-32.
- [Cleveland 1988] W.S. Cleveland, S.J. Delvin (1988), "Locally weighted regression: An approach to regression analysis by local fitting", *Journal of the American Statistical Association*, 83:596-610.
- [Crites 1995] B. Crites (1995), "A Case Study from the Elevator Dispatching Domain", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Dayan 1992] P. Dayan (1992), "The Convergence of TD(λ) for General λ ", *Machine Learning*, 8:341-362.
- [Dayan 2001] P. Dayan, L.F. Abbott (2001), *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, MIT Press, Cambridge, MA.
- [Deng 1997] Z.D. Deng, D.P. Kwok (1997), "Dynamic neural network control through fuzzy Q-learning algorithms", *IEEE International Conference on Intelligent Processing Systems, 1997 (ICIPS '97)*, 1:381-386.
- [Ding 2003] Y. Ding (2003), *Reinforcement Learning in Optimal Ventilator Control*, MSc Dissertation, Singapore-MIT Alliance.
- [Doya 2000] K. Doya (2000), "Reinforcement learning in continuous time and space", *Neural Computation*, 12:219-245.
- [Dreyfus 1995] S. Dreyfus (1995), "AHC for Stochastic Minimum Cost Path Problems Using Neural Net Function Approximators", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Dubios 2000] D. Dubios, H. Prade (2000), *Fundamentals of Fuzzy sets*, Kluwer Academic Publishers.
- [Duda 1973] R. Duda, P. Hart (1973), *Pattern Classification and Scene Analysis*, Wiley.

- [Freedman 1983] D. Freedman (1983), *Markov Chains*, Springer-Verlag.
- [Fu 2002a] X. Fu, L. Wang (2002), "Rule extraction from an RBF classifier based on class-dependent features", *CEC 2002*, 2:1916-1921.
- [Fu 2002b] X. Fu, L. Wang (2002), "A GA-based novel RBF classifier with class-dependent features", *CEC 2002*, 2:1890-1894.
- [Fu 2003] X. Fu, L. Wang (2003), "Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance", *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):399-409.
- [Furlanello 2003] C. Furlanello, M. Serafini, S. Merler, G. Jurman (2003), "An accelerated procedure for recursive feature ranking on microarray data", *Neural Networks*, 16:641-648.
- [Gan 1999] Q. Gan, C.J. Harris (1999), "Fuzzy local linearization and local basis function expansion in nonlinear system modeling", *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(4):559-565.
- [Gan 2001] Q. Gan, C.J. Harris (2001), "A hybrid learning scheme combining EM and MASMOD algorithms for fuzzy local linearization modeling", *IEEE Trans. Neural Networks*, 12(1): 43-53.
- [Geman 1992] S. Geman, E. Bienenstock, R. Doursat (1992), "Neural Networks and the Bias/Variance Dilemma", *Neural Computation*, 4:1-58.
- [Glorennec 1994] P.Y. Glorennec (1994), "Fuzzy Q-learning and dynamical fuzzy Q-learning", *Proceedings of the Third IEEE Conference on Fuzzy Systems*, 1:474-479.
- [Glorennec 1997] P.Y. Glorennec, L. Jouffe (1997), "Fuzzy Q-learning", *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, 2:659-662.
- [Goldberg 1989] D.E. Goldberg (1989), *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley Co., MA.
- [Gordon 1995a] G. Gordon (1995), "Stable function approximation in dynamic programming", *ICML 1995*, 261-268.
- [Gordon 1995b] G. Gordon (1995), "Online Fitted Reinforcement Learning", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Hardle 1990] W. Hardle (1990), *Applied Nonparametric Regression*, Cambridge University Press.
- [Harmon 1996] M.E. Harmon (1996), "Reinforcement learning: a tutorial", <http://citeseer.ist.psu.edu/harmon96reinforcement.html>.
- [Harris 1999] C.J. Harris, Z.Q. Wu, Q. Gan (1999), "Neurofuzzy state estimators and their applications", *Ann. Rev. in Control*, 23:149-158.

- [Harris 2001] C.J. Harris, Q. Gan (2001), "State estimation and multi-sensor data fusion using data-based neurofuzzy local linearisation process models", *Information Fusion*, 2(1):17-29.
- [Hsu 2002] C.N. Hsu, H.J. Huang, S. Dietrich (2002), "The ANNIGMA-Wrapper Approach to Fast Feature Selection for Neural Nets", *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 32(2): 207-212.
- [Hull 1997] J.C. Hull (1997), *Options, Futures, and other Derivatives*, 3rd ed., Prentice Hall.
- [Jaakkola 1994] T. Jaakkola, M.I. Jordan, S.P. Singh (1994), "On the convergence of stochastic iterative dynamic programming algorithms", *Neural Computation*, 6:1185-1201.
- [Jang 1992] J.S. Jang (1992), *Neuro-Fuzzy Modeling: Architecture, Analyses and Applications*, PhD Dissertation, Department of EE and CS, U. of California, Berkeley.
- [Jang 1993] J.S. Jang (1993), "ANFIS: Adaptive-network-based fuzzy inference system", *IEEE Trans. on System, Man and Cybernetics*, 23(3):665-685.
- [Jang 1996] J.S. Jang (1996), "Input Selection for ANFIS Learning", in *Proceedings of the IEEE International Conference on Fuzzy Systems*, New Orleans, 1996.
- [Jensen 2004] R. Jensen, Q. Shen (2004), "Fuzzy-rough attribute reduction with application to web categorization", *Fuzzy Sets and Systems*, 141(3):469-485.
- [Jensen 2005] R. Jensen, Q. Shen (2005), "Fuzzy-rough data reduction with ant colony optimization", *Fuzzy Sets and Systems*, 149(1):5-20.
- [Jiang 2003] H. Jiang (2003), *Fuzzy CMAC Implementing CRI*, Final Year Project Report, School of Computer Engineering, Nanyang Technological University.
- [Johansen 2000] T.A. Johansen, R. Shorten; R. Murray-Smith (2000), "On the interpretation and identification of dynamic Takagi-Sugeno fuzzy models", *IEEE Trans. on Fuzzy Systems*, 8(3):297-313.
- [Johansen 2003] T.A. Johansen, R. Babuska (2003), "Multiobjective identification of Takagi-Sugeno fuzzy models", *IEEE Trans. on Fuzzy Systems*, 11(6):847-860.
- [John 1994] G.H. John, R. Kohavi, K. Pfleger (1994), "Irrelevant Features and the Subset Selection Problem", *Proceedings of the 11th International Conference on Machine Learning*, 121-129, Morgan Kaufmann.
- [Jouffe 1997] L. Jouffe (1997), "Ventilation control learning with FACL", *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, 3:1719-1724.

- [Jouffe 1998] L. Jouffe (1998), "Fuzzy inference system learning by reinforcement methods", *IEEE Trans. on SMC-Part C, Applications and Reviews*, 28(3): 338-355.
- [Kaelbling 1996] L.P. Kaelbling, L.M. Littman, A.W. Moore (1996), "Reinforcement learning: a survey", *Journal of Artificial Intelligence Research*, 4:237-285.
- [Kaufmann 1985] A. Kaufmann, M.M. Gupta (1985), *Introduction to Fuzzy Arithmetic*, Reinhold, NY.
- [Kim 1999] M.S. Kim, G.G. Hong, J.J. Lee (1999), "Online fuzzy Q-learning with extended rule and interpolation technique", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999 (IROS '99)*, 2:757-762.
- [Kira 1992a] K. Kira, L.A. Rendell (1992), "The feature selection problem: Traditional methods and a new algorithm", *Tenth National Conference on Artificial Intelligence*, MIT Press, 129-134.
- [Kira 1992b] K. Kira, L.A. Rendell (1992), "A practical approach to feature selection", *Proceedings of the Ninth International Conference on Machine Learning*, Morgan Kaufmann.
- [Kohavi 1997] R. Kohavi, G.H. John (1997), "Wrappers for Feature Subset Selection", *Artificial Intelligence*, 97(1-2):273-324.
- [Kohonen 1982] T. Kohonen (1982), "Self-organized formation of topologically correct feature maps", *Biological Cybernetics*, 43:59-69.
- [Konda 2003] V.R. Konda and J.N. Tsitsiklis (2003), "Actor-Critic Algorithms", *SIAM Journal on Control and Optimization*, 42(4):1143-1166.
- [Kononenko 1994] I. Kononenko (1994), "Estimating attributes: Analysis and Extensions of RELIEF", in F. Bergadano & L. D. Raedt, eds., *Proceedings of the European Conference on Machine Learning*.
- [Kwak 2002] N. Kwak, C.H. Choi (2002), "Input Feature Selection for Classification Problems", *IEEE Trans. on Neural Networks*, 13(1): 143-159.
- [Laar 1997] P. Laar, S. Gielan, T. Heskes (1997), "Input Selection with Partial Retraining", *ICANN 1997*, 469-474.
- [Laar 1999] P. Laar, S. Gielan, T. Heskes (1999), "Partial retraining: a new approach to input relevance determination", *International Journal of Neural Systems*, 9:75-85.
- [Langley 1994] P. Langley (1994), "Selection of relevant features in machine learning", in *AAAI Fall Symposium on Relevance*, 140-144, AAAI Press.
- [Lee 1990a] C.C. Lee (1990), "Fuzzy Logic in Control Systems: Fuzzy Logic Controller: Part I", *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2): 404-418.

- [Lee 1990b] C.C. Lee (1990), "Fuzzy Logic in Control Systems: Fuzzy Logic Controller: Part II", *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2): 419-435.
- [Lee 2001] H.M. Lee, C.M. Chen, J.M. Chen, Y.L. Jou (2001), "An Efficient Fuzzy Classifier with Feature Selection Based on Fuzzy Entropy", *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 31(03):426-432.
- [Lin 1992] L.J. Lin (1992), "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching", *Machine Learning*, 8: 293-321.
- [Lin 1995] C.T. Lin, Y.C. Lu (1995), "A Neural Fuzzy System with Linguistic Teaching Signals", *IEEE Trans. on Fuzzy Systems*, 3(2):169-189.
- [Lin 1996a] C.J. Lin, C.T. Lin (1996), "Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network", *IEEE Trans. on Neural Networks*, 7(3):709-731.
- [Lin 1996b] C.T. Lin, C.S. Lee (1996), *Neural fuzzy systems: A neuro-fuzzy synergism on intelligent system*, Prentice-Hall PTR, NJ.
- [Lin 1998] C.T. Lin, M.C. Kan (1998), "Adaptive fuzzy command acquisition with reinforcement learning", *IEEE Trans. on Fuzzy Systems*, 6(1):102-121.
- [Littman 1995] M. Littman (1995), "Solving Partially Observable Markov Decision Processes via VFA", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Littman 1996] M. Littman, C. Szepesvari (1996), "A generalized reinforcement-learning model: Convergence and applications", In Saitta, L. (ed.), *Proceedings of the Thirteenth International Conference on Machine Learning*, 310-318.
- [Liu 1998] H. Liu, H. Motoda (1998), *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers.
- [Mahadevan 1996] S. Mahadevan (1996), "Average reward reinforcement learning: foundations, algorithms, and empirical results", *Machine Learning*, 22:159-196.
- [Mamdani 1975] E.H. Mamdani, S. Assilian (1975), "An experiment in linguistic synthesis with a fuzzy logic controller", *Int. J. Machine Studies*, 7(1).
- [Männle 1999] M. Männle (1999), "Identifying Rule-Based TSK Fuzzy Models", *Proceedings of EUFIT*, Aachen, Germany, September 1999.
- [Mao 2002] K.Z. Mao (2002), "Fast Orthogonal Forward Selection Algorithm for Feature Subset Selection", *IEEE Trans. on Neural Networks*, 13(5):1218-1224.
- [Moody 1992] J. Moody, J. Utans (1992), "Principled architecture selection for neural networks: Application to corporate bond rating prediction", in J. E. Moody, S. J. Hanson and R. P. Lippmann, eds, *Advances in Neural*

Information Processing Systems 4, 683-690, Morgan Kaufmann Publishers.

- [Moody 1995] J. Moody, J. Utans (1995), "Architecture selection strategies for neural networks: Application to corporate bond rating prediction", in A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, 277-300, John Wiley & Sons.
- [Moore 1995] A.W. Moore (1995), "Value Function Approximation", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Moore 1997] A.W. Moore, J. Schneider, K. Deng (1997), "Efficient locally weighted polynomial regression predictions", *ICML 1997*.
- [Mouzouris 1997a] G.C. Mouzouris, J.M. Mendel (1997), "Nonsingleton fuzzy logic systems: theory and application", *IEEE Trans. Fuzzy Systems*, 5(1):56-71.
- [Mouzouris 1997b] G.C. Mouzouris, J.M. Mendel (1997), "Dynamic non-singleton fuzzy logic systems for nonlinear modeling", *IEEE Trans. Fuzzy Systems*, 5(2):199-208.
- [Mozer 1989] M.C. Mozer, P. Smolensky (1989), "Using relevance to reduce network size automatically", *Connection Science*, 1(1):3-16.
- [Nakanishi 1993] H. Nakanishi, I.B. Turksen, M. Sugeno (1993), "A review and comparison of six reasoning methods", *Fuzzy Sets and Systems*, 57:257-294.
- [Parsley 1997] M. Parsley (1997), "Exotics enter the mainstream", *Euromoney*, 127-130.
- [Peng 1996] J. Peng, R.J. Williams (1996), "Incremental multi-step Q-learning", *Machine Learning*, 22:283-290.
- [Petersen 1998] M.E. Petersen, J.L. Talmon, A. Hasman, A.W. Amberg (1998), "Assessing the importance of features for multi-layer perceptrons", *Neural Networks*, 11:623-635.
- [Puterman 1994] M.L. Puterman (1994), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons.
- [Quah 2004a] K.H. Quah, C. Quek (2004), "Structural credit assignment in reinforcement learning with fuzzy modelling: preliminary results", *Technical Report C2i-TR-04/001*, Centre for Computational Intelligence, Nanyang Technological University.
- [Quah 2004b] K.H. Quah, C. Quek (2004), "FITSK: Online local learning with generic fuzzy input Takagi-Sugeno-Kang fuzzy framework for nonlinear system estimation", *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):166-178, 2006.
- [Quah 2004c] K.H. Quah, C. Quek (2004), "FITSK-RL: Function approximation with the fuzzy input Takagi-Sugeno-Kang model for a generic reinforcement learning framework", *submitted to Machine Learning*.

- [Quah 2004d] K.H. Quah, C. Quek (2004), "A Backward-Replay Approach to ICU Ventilator Control with Reinforcement Learning", *Technical Proposal*, Centre for Computational Intelligence, Nanyang Technological University.
- [Quinlan 1986] J.R. Quinlan (1986), "Induction of decision trees", *Machine Learning*, 1:81-106, Reprinted in Shavlik and Dietterich (eds.) *Readings in Machine Learning*.
- [Quinlan 1993] J.R. Quinlan (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California.
- [Rees 1999] S.E. Rees, S. Andreassen, M. Freundlich, C. Morgan, E.R. Carson, P. Thorgaard (1999), "Selecting Ventilator Settings using INVENT, a System including Physiological Models and Penalty Functions", *Workshop on Computers in Anaesthesia and Intensive Care: Knowledge-Based Information Management, AIMDM 1999*.
- [Rumelhart 1986a] D.E. Rumelhart, G.E. Hinton, R.J. Williams (1986), "Learning internal representations by error propagation", *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol 1:318-362, MIT Press.
- [Rumelhart 1986b] D.E. Rumelhart, G.E. Hinton, R.J. Williams (1986), "Learning representations by back-propagating errors", *Nature*, 323:533-536.
- [Rummery 1994] G.A. Rummery, M. Niranjana (1994), "On-line Q-learning using connectionist systems", *Technical Report CUED/F-INFENG/TR166*, Cambridge University Engineering Dept.
http://svr-www.eng.cam.ac.uk/reports/svr-ftp/rummery_tr166.ps.Z
- [Rummery 1995] G.A. Rummery (1995), *Problem solving with reinforcement learning*, PhD Thesis, University of Cambridge.
- [Rutledge 1993] G.W. Rutledge, G.E. Thomsen, B.R. Farr, M.A. Tovar, J.X. Polaschek, I.A. Beinlich, L.B. Sheiner, L.M. Fagan (1993), "The Design and Implementation of a Ventilator-Management Advisor", *Artificial Intelligence in Medicine*, 5(1):67-82.
- [Schwartz 1993] A. Schwartz (1993), "A reinforcement learning method for maximizing undiscounted rewards", *ICML 1993*, 298-305.
- [Setiono 1997] R. Setiono, H. Liu (1997), "Neural-network feature selector", *IEEE Trans. on Neural Networks*, 8(3): 654-659.
- [Setiono 2002] R. Setiono, W.K. Leow, J.M. Zurada (2002), "Extraction of Rules from Artificial Neural Networks for Nonlinear Regression", *IEEE Trans. on Neural Networks*, 13(3):564-577.
- [Shen 1999] Q. Shen, A. Chouchoulas (1999), "Combining rough sets and data-driven fuzzy learning for generation of classification rules", *Pattern Recognition*, 32(12):2073-2076.

- [Shen 2000] Q. Shen, A. Chouchoulas (2000), "A modular approach to generating fuzzy rules with reduced attributes for the monitoring of complex systems", *Engineering Applications of Artificial Intelligence*, 13(3):263-278.
- [Shen 2001a] Q. Shen, A. Chouchoulas (2001), "Rough set-based dimensionality reduction for supervised and unsupervised learning", *Int. J. Appl. Math. Comput. Sci.*, 11(3):583-601.
- [Shen 2001b] Q. Shen, A. Chouchoulas (2001), "FuREAP: a fuzzy-rough estimator of algae populations", *Artificial Intelligence in Engineering*, 15(1):13-24.
- [Shen 2002] Q. Shen, A. Chouchoulas (2002), "A rough-fuzzy approach for generating classification rules", *Pattern Recognition*, 35(11):2425-2438.
- [Shen 2004] Q. Shen, R. Jensen (2004), "Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring", *Pattern Recognition*, 37(7):1351-1363.
- [Singh 1994] S. P. Singh (1994), *Learning to solve Markovian Decision Processes*, PhD Dissertation, University of Massachusetts, Amherst, MA.
- [Singh 1996] S.P. Singh, R.S. Sutton (1996), "Reinforcement learning with replacing eligibility traces", *Machine Learning*, 22:123-158.
- [Smart 2000] W.D. Smart, L.P. Kaelbling (2000), "Practical reinforcement learning in continuous spaces", *ICML 2000*, 903-910.
- [Strogatz 1994] S.H. Strogatz (1994), *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, Addison-Wesley.
- [Sugeno 1988] M. Sugeno, K.T. Kang (1988), "Structure identification of fuzzy model", *Fuzzy Sets and Systems*, 28.
- [Suh 1997] I.H. Suh, J.H. Kim, F.C.H. Rhee (1997), "Fuzzy-Q learning for autonomous robot systems", *International Conference on Neural Networks*, 3:1738-1743.
- [Sun 1996] Z. Sun, Z. Deng (1996), "A fuzzy neural network and its application to controls", *Artificial Intelligence in Engineering*, 10:311-315.
- [Sutton 1984] R.S. Sutton (1984), *Temporal Credit Assignment in Reinforcement Learning*, PhD Dissertation, University of Massachusetts, Amherst, MA.
- [Sutton 1988] R.S. Sutton (1988), "Learning to predict by the method of temporal differences", *Machine Learning*, 3(1):9-44.
- [Sutton 1989] R.S. Sutton (1989), "Implementation details of the TD(λ) procedure for the case of vector predictions and backpropagation", *Technical Report TN87-509.1*, GTE Laboratories.

- [Sutton 1995] R.S. Sutton (1995), "On the Virtues of Linear Learning and Trajectory Distributions", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Sutton 1996a] R.S. Sutton, A.G. Barto (1996), "Reinforcement Learning I: Introduction; II: Learning Values; III: Applications", *Course Notes*, University of Massachusetts.
ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/IntroRL/RL1.ps,RL2.ps,RL3.ps.
- [Sutton 1996b] R.S. Sutton (1996), "Generalization in reinforcement learning: Successful examples using sparse coarse coding", In D. Touretzky, M. Mozer, and M. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, 8:1038-1044, MIT Press.
- [Sutton 1998] R.S. Sutton, A.G. Barto (1998), *Reinforcement Learning: An introduction*, The MIT Press, Cambridge, MA.
- [Sutton 1999] R.S. Sutton (1999), "Reinforcement Learning", In Rob Wilson and Frank Keil (Eds.), *The MIT Encyclopedia of the Cognitive Sciences*, MIT Press.
- [Tadepalli 1998] P. Tadepalli, D.K. Ok (1998), "Model-based average reward reinforcement learning", *Artificial Intelligence*, 100:177-224.
- [Takagi 1985] T. Takagi, M. Sugeno (1985), "Fuzzy identification of systems and its application to modeling and control", *IEEE Trans. on Systems, Man and Cybernetics*, 15(1):116-132.
- [Tan 2003] T.Z. Tan (2003), *PACL: A Class of Novel Complimentary Fuzzy Neural Network*, Final Year Project Report, School of Computer Engineering, Nanyang Technological University.
- [Ting 2001] C.W. Ting (2001), *Embedded Audio Detection Software*, Honours Year Project report, School of Computer Engineering, Nanyang Technological University.
- [Ting 2002] C.W. Ting (2002), *Fuzzy associative memory architecture*, PhD First year confirmation report, School of Computer Engineering, Nanyang Technological University.
- [Tsitsiklis 1996] J.N. Tsitsiklis, B. Van Roy (1996), "Feature-Based Methods for Large Scale Dynamic Programming", *Machine Learning*, 22:59-94.
- [Tsitsiklis 1997] J.N. Tsitsiklis, B. Van Roy (1997), "An analysis of temporal-difference learning with function approximation", *IEEE Trans. on Automatic Control*, 42(5):674-690.
- [Tsitsiklis 1999] J.N. Tsitsiklis, B. Van Roy (1999), "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives", *IEEE Trans. Automatic Control*, 44(10):1840-1851.
- [Tung 2002] W.L. Tung, C. Quek (2002), "GenSoFNN: A generic self-organizing fuzzy neural network", *IEEE Trans. on Neural Networks*, 13(5):1075-1086.

- [Tung 2003] W.L. Tung (2003), *A Generalized Framework for Fuzzy Neural Architecture*, PhD thesis, School of Computer Engineering, Nanyang Technological University, Singapore.
- [Tung 2005] W.L. Tung, C. Quek (2005), "GenSo-FDSS: A Neural-Fuzzy Decision Support System for Pediatric ALL Cancer Subtype Identification Using Gene Expression Data", *Artificial Intelligence in Medicine*, 33(1):61-88.
- [Utans 1991] J. Utans, J. Moody (1991), "Selecting neural network architectures via the prediction risk: Application to corporate bond rating prediction", in *Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street*, IEEE Computer Society Press.
- [Van Roy 1998] B. Van Roy (1998), *Learning and value function approximation in complex decision processes*, PhD Dissertation, Massachusetts Institute of Technology.
- [Wang 1998] C.S. Wang, D. Shaw, K.S. Jih (1998), "An intelligent control system for ventilators", *Medical Engineering & Physics*, 20:534-542.
- [Wang 2001] L. Wang, K.K. Teo, Z. Lin, "Predicting time series with wavelet packet neural networks", *IJCNN 2001*, 1593-1597.
- [Watkins 1989] C.J.C.H. Watkins (1989), *Learning with delayed rewards*, PhD dissertation, Psychology Department, Cambridge University, England.
- [Watkins 1992] C.J.C.H. Watkins, P. Dayan (1992), "Technical Note: Q-Learning", *Machine Learning*, 8:279-292.
- [White 1993] D.J. White (1993), *Markov Decision Processes*, John Wiley & Sons, New York.
- [Widrow 1960] B. Widrow, M. E. Hoff (1960), "Adaptive switching circuits", *1960 IRE WESCON Convention Record*, New York:IRE, 96-104.
- [Williams 1992] R.J. Williams (1992), "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning", *Machine Learning*, 8:229-256.
- [Xu 2002] X. Xu, H. He, D. Hu (2002), "Efficient reinforcement learning using recursive least-squares methods", *J. Artificial Intelligence Research*, 16:259-292.
- [Yan 2001] X.W. Yan, Z.D. Deng, Z.Q. Sun (2001), "Genetic Takagi-Sugeno fuzzy reinforcement learning", *Proceedings of the 2001 IEEE International Symposium on Intelligent Control, 2001 (ISIC '01)*, 67-72.
- [Yen 1998] J. Yen, L. Wang, C.W. Gillespie (1998), "Improving the interpretability of TSK fuzzy models by combining global learning and local learning", *IEEE Trans. on Fuzzy Systems*, 6(4):530-537.

- [Yen 1999] J. Yen, R. Langari (1999), *Fuzzy logic: intelligence, control, and information*, Prentice Hall, NJ.
- [Zadeh 1965] L.A. Zadeh (1965), "Fuzzy Sets", *Information and Control*, 8(3):338-353.
- [Zhang 1995] W. Zhang, T. Dietterich (1995), "Value Function Approximations and Job-Shop Scheduling", *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference 1995*.
- [Zhou 1996] R.W. Zhou, C. Quek (1996), "POPFNN: a Pseudo Outer-Product based Fuzzy Neural Network", *Neural Networks*, 9(9):1569-1581.
- [Zimmermann 1985] H.J. Zimmermann (1985), *Fuzzy Set Theory - and its Applications*, Kluwer Academic Publishers.

Author's Publications

K.H. Quah, C. Quek and G. Leedham, "Reinforcement Learning combined with a Fuzzy Adaptive Learning Control Network (FALCON-R) for Pattern Classification", *Pattern Recognition*, 38(4):513-526, 2005.

K.H. Quah, C. Quek, "FITSK: Online Local Learning with Generic Fuzzy Input Takagi-Sugeno-Kang Fuzzy Framework for Nonlinear System Estimation", *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):166-178, Feb. 2006.

K.H. Quah, C. Quek, "Maximum Reward Reinforcement Learning: A Non-Cumulative Reward Criterion", *Expert Systems with Applications*, 2006 (in press).

K.H. Quah, C. Quek and G. Leedham, "Pattern Classification Using Fuzzy Adaptive Learning Control Network and Reinforcement Learning", *ICONIP 2002*.

K.H. Quah, C. Quek, "Structural credit assignment in reinforcement learning with fuzzy modelling: preliminary results", *Technical Report C2i-TR-04/001*, Centre for Computational Intelligence, Nanyang Technological University, 2004.

K.H. Quah, C. Quek, "FITSK-RL: Function Approximation with the Fuzzy Input Takagi-Sugeno-Kang Model for a Generic Reinforcement Learning Framework", *submitted to Machine Learning*, 2004 (1st revision).

K.H. Quah, C. Quek, "MCES: A Novel Monte Carlo Evaluative Selection Approach for Objective Feature Selections", *submitted to IEEE Trans. Neural Networks*, 2004 (1st revision).

PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®

Pattern Recognition 38 (2005) 513–526

**PATTERN
RECOGNITION**

THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Reinforcement learning combined with a fuzzy adaptive learning control network (FALCON-R) for pattern classification

K.H. Quah, C. Quek, G. Leedham*

Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, N4 #2A-32 Nanyang Avenue, Nanyang 639798, Singapore

Received 17 October 2003; received in revised form 8 June 2004; accepted 13 August 2004

Abstract

Reinforcement learning has been widely-used for applications in planning, control, and decision making. Rather than using instructive feedback as in supervised learning, reinforcement learning makes use of evaluative feedback to guide the learning process. In this paper, we formulate a pattern classification problem as a reinforcement learning problem. The problem is realized with a temporal difference method in a FALCON-R network. FALCON-R is constructed by integrating two basic FALCON-ART networks as function approximators, where one acts as a critic network (fuzzy predictor) and the other as an action network (fuzzy controller). This paper serves as a guideline in formulating a classification problem as a reinforcement learning problem using FALCON-R. The strengths of applying the reinforcement learning method to the pattern classification application are demonstrated. We show that such a system can converge faster, is able to escape from local minima, and has excellent disturbance rejection capability.

© 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Falcon-ART; Reinforcement learning; Classification system; Dynamic programming; Noise tolerance; Performance evaluation; Fuzzy rules; Disturbance rejection; Local minima; Critic and action networks; Temporal difference prediction.

1. Introduction

Reinforcement learning is another method of learning in machine intelligence. It has the ability to make use of evaluative feedback from the environment to guide the learning process. It has been widely applied to the application of planning, control and decision making [1,2]. The learning classifier systems [3] was first established as a form of reinforcement learning systems that include association and value functions ([4], p. 20). Peng and Bhanu [5,6] applied reinforcement learning techniques to search for

optimal parameters in segmentation and feature extraction stages of object recognition. The classification or model matching stage of object recognition was defined as an episodic undiscounted reinforcement learning framework, where active classifiers are viewed as policies to map states to the actions [7]. These foundations support the formulation of reinforcement learning into classifier systems.

In this paper, a reinforcement learning system, implemented using the temporal difference method [1] is realized in a fuzzy adaptive learning control network (FALCON-R) [8,9] for the application of pattern classification. The purpose of this experiment is not to show that reinforcement learning outperforms supervised learning methods in pattern classification. Instead, by adding reinforcement learning ability to a supervised learning technique (FALCON-ART) [10], forming the FALCON-R network, we identify

* Corresponding author. Tel.: +65 6 790 6250;
fax: +65 6 792 6559.

E-mail address: asgledham@ntu.edu.sg (G. Leedham).

and demonstrate some strengths over the supervised learning method. Furthermore, our purpose is to provide guidelines for formulating an AHC reinforcement learning model [11] as a classifier.

2. Reinforcement learning

Reinforcement Learning is an approach to machine intelligence. Instead of applying supervised or unsupervised methods to a learning problem, the reinforcement learning agent dynamically interacts with the environment. With evaluative feedback passing back from the environment, a value function is formulated to guide the action towards the desired goal state.

Reinforcement learning is itself a problem requiring further research. Recently, an approach that combined two disciplines (dynamic programming and supervised learning) to realize the reinforcement learning problem was proposed [12] and successfully demonstrated to solve problems that neither discipline can address individually. Dynamic Programming [2] is a field of mathematics that has traditionally been used to solve problems of optimization and control. However, dynamic programming can only address problems of a limited size and complexity. Supervised learning is a general method for training a parameterized function approximator, such as a neural network, to represent functions. However, supervised learning requires a large set of sample input–output pairs from the function to be learned. Instead of learning from direct input–output matching, a reinforcement learning agent learns by directly interacting with the environment, collecting the evaluative feedback it encounters, and finding the optimal actions that lead to maximal long term rewards. Reinforcement learning combines the fields of dynamic programming and supervised learning to yield a powerful machine-learning system.

If $r(t)$ is the reinforcement signal available at time step t and is caused by the input and actions chosen at time step $(t - 1)$ or even affected by earlier input and actions. Then, in general, the reinforcement signal $r(t)$, can be in one of the following three forms:

1. A two-valued number, $r(t) \in \{-1, 1\}$ or $\{0, 1\}$, such that $r(t) = 1$ means “success” and $r(t) = -1$ (or 0) means ‘failure’.
2. A multivalued discrete number in the range of $[-1, 1]$ or $[0, 1]$, for example, $r(t) \in \{-1, -0.5, 0.5, 1\}$ that corresponds to different discrete degrees of failure or success.
3. A real number, $r(t) \in [-1.0, 1.0]$ or $[0.0, 1.0]$ that represents a more detailed and continuous degree of failure or success.

There are three classes of reinforcement learning problems Hertz et al. [13].

1. The simplest case, where the reinforcement signal is always the same for a given input–output pair.

2. In a stochastic environment, where a particular input–output pair determines only the probability of positive reinforcement.
3. The most general case, where the environment is itself governed by a complicated dynamic process, and both the reinforcement signal and input patterns may depend on past network output. The reinforcement signal may appear after a long delay.

It is this third, or general case, which we wish to examine in more detail.

2.1. The general case of temporal difference methods

The general case of delayed reinforcement learning is characterized by problems in which there is a long time delay between a reinforcement signal and the actions that caused it. In such cases, a temporal credit assignment problem results because of the need to assign credit or blame for an eventual success or failure, to each step individually in a long sequence. A widely used approach in developing algorithms for such tasks is to combine an immediate-reinforcement network with an adaptive predictor or critic to perform the task of predicting reinforcement based on the use of temporal difference (TD) methods [1].

Prediction of infinite discounted cumulative outcomes can be achieved as follows:

Given that the observation–outcome sequences are of the form $x_1, x_2, \dots, x_t, \dots, x_m, z$, where each x_t is an input vector available at time step t from the environment and z is the external reinforcement signal available at time step $(m + 1)$. If p_t denotes the output of the critic network at time t , it is a function of the network’s input x_t and the network’s adjustable parameters (weight) w_t . Thus, p_t can be denoted as $p(x_t, w_t)$.

Let r_t be the actual cost incurred between time steps $(t - 1)$ and t . In this case, p_{t-1} predicts $z_{t-1} = \sum_{k=0}^{\infty} \xi^k r_{t+k} = r_t + \xi p_t$, where the discount rate parameter ξ , $0 \leq \xi < 1$, determines the extent to which it is a short-range or a long-range prediction. This method is used for prediction problems in which exact success or failure may never become completely known. In this case, the prediction error is $(r_t + \xi p_t) - p_{t-1}$ and the learning rule is

$$\Delta w_t = \eta(r_t + \xi p_t - p_{t-1}) \sum_{k=1}^{t-1} \lambda^{t-k-1} \nabla_w p_k, \quad (1)$$

where

- $p_m = z$, is the prediction at time step m ,
- $0 \leq \lambda \leq 1$, λ is the recency weighting factor with which alternations in the predictions of the observation vectors occurring k steps in the past are weighted by λ^k , and
- η is the learning rate.

2.2. Formulation of the reinforcement signal for the classifier

In order to realise an effective reinforcement learning system, a “wise” external reinforcement signal must be formulated in order to represent the actual state of the learning system. The reinforcement signal is system dependent. A formulation suitable for one system may not be suitable for another system.

In this paper, two types of external reinforcement signal are formulated and described below, where “success” is assigned as 0 and “failure” is assigned as -1 .

One reinforcement signal represents the general case, and is suitable for most reinforcement learning networks. The second reinforcement signal is a specific formulation, which is suitable for use in a classifier type of reinforcement learning network. The performance of the second formulation is better than the first one because it is more system specific (see Section 4).

2.2.1. General formulation of the external reinforcement signal

The number of external reinforcement signals is dependent on the number of outputs of the action network. Every external reinforcement signal is responsible for predicting the outcome of an output node of the action network. (sentence deleted) Every output node in the critic network produces its own predicted reinforcement signal for its corresponding action output.

For this formulation, a cutoff value is needed, whenever the action network produces an action that is uncorrelated with the expected action, a “failure” is given as the external reinforcement signal, else a “success” is given. The amount of correlation between the expected action and the actual action is given by this cutoff value.

The algorithm for the general formulation of the external reinforcement signal is:

Step 1: Retrieve the current output(s) y_{actual} of the action network (actual output) for the current training input data.

Step 2: Retrieve the desired output(s) $y_{desired}$ from the training data.

Step 3: Compute the correlation expressed in terms of the output signal using the formula

$$\text{Correlation} = \frac{y_{actual} - y_{desired}}{y_{desired}} \quad (2)$$

Step 4: If $|\text{Correlation}| < \text{cutoff}$, then “success” is generated, else “failure” is generated for the external reinforcement signal.

Step 5: Repeat for every output of the action network. Every output of the action network has its own external reinforcement signal at the critic network.

This is a general formulation of the external reinforcement signal, which is adaptable to other systems. When the

actual success or failure of the system is hard to predict, this formulation of the external reinforcement signal gives a correlated version of success or failure depending on the expected and actual actions of the system.

2.2.2. Formulation of an external reinforcement signal based on the classification test

This formulation of the external reinforcement signal is system dependent. It can be used in reinforcement training for pattern classification problems. There is only a single external reinforcement signal supplied by the critic network. The output of the critic network indicates whether the action produced by the action network causes a failure or a success.

The algorithm for the formulation of an external reinforcement signal for pattern classification is:

Step 1: Retrieve the current output(s) of the action network (actual output) for the current training input data.

Step 2: Retrieve the pattern class to which the output set belongs (i.e., class 1, 2, or 3 for the Iris set used later in this study).

Step 3: Retrieve the expected class of the pattern data.

Step 4: When the classification is correct, a “success” is given, otherwise a “failure” is given.

The external reinforcement signal formulated using this approach represents the whole system and whether it is a success or failure. The outputs of every action network to be trained in the subsequent steps are based on the single predicted reinforcement signal of this critic network.

This formulation of external reinforcement signal can be generalized for any m class classification problem. Where a “success” is given for successful classification, and “failure” given whenever a mis-classification occurs.

3. FALCON-R classifier

The formulation of a reinforcement method for solving a classification problem is realized with a FALCON-R [8,9] network. This section describes the FALCON-R network, its construction from the basic building blocks of FALCON-ART networks [10] and its formulation into a classifier.

3.1. FALCON-ART

A FALCON-ART network is trained using a fuzzy learning algorithm. Initially, there are no hidden nodes (i.e., no membership functions or fuzzy rules). They are created and begin to grow as learning proceeds. The FALCON-ART network can also dynamically partition the input–output spaces, tune activation (membership) functions, and find the appropriate network connections (fuzzy rules). The problem of space partitioning from numerical training data is basically a clustering problem. The FALCON-ART network applies fuzzy adaptive resonance theory (fuzzy ART)

Fig. 1. Structure of the FALCON-ART network.

to perform fuzzy clustering in the input–output spaces using fuzzy hyperboxes. It dynamically derives the appropriate fuzzy logic rules by associating input clusters (input hyperboxes) with output clusters (output hyperboxes). The back-propagation learning scheme is subsequently used to tune the input–output membership functions. Hence, the FALCON-ART network combines a back-propagation algorithm for parameter learning and a fuzzy ART for structure learning.

The FALCON-ART system has a five-layer structure as shown in Fig. 1. Layer 1 nodes are input nodes and represent input linguistic variables. Layer 5 nodes are output nodes and represent output linguistic variables. The nodes of layer 2 and 4 are term nodes that act as membership functions representing the terms of the respective input and

output linguistic variables. Each layer 3 node is a rule node representing one fuzzy logical rule. Together, all the layer 3 nodes form the fuzzy rule base. Links between layers 3 and 4 function as a connectionist inference engine. Layer 3 links define the pre-conditions of the rule node, and layer 4 links define the post-conditions of the rule nodes. FALCON-ART uses the technique of complementary coding from Fuzzy ART [14] to normalize the input–output training vectors.

The fuzzy reasoning process in the FALCON-ART network is illustrated in Fig. 2. It shows a graphical interpretation of the center of area defuzzification method for a two input and two output case. Three input hyperboxes (IH_1, IH_2, IH_3) are formed in the input space and two output hyperboxes (OH_1, OH_2) are formed in the output space. These hyperboxes are defined by the weights u_{ij}, v_{ij} , and u'_{ij}, v'_{ij} .

Fig. 2. The fuzzy reasoning process in the FALCON-ART model.

The three fuzzy rules indicated in Fig. 2 are

IF x is IH_1 then y is OH_1 (rule 1);

IF x is IH_2 then y is OH_1 (rule 2);

IF x is IH_3 then y is OH_3 (rule 3);

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$.

3.2. Formulation of FALCON-R as a classifier

The FALCON-R is a learning system that can automatically and dynamically construct a fuzzy control and decision system through a reward-penalty signal or through very simple fuzzy feedback information. Moreover, there is a possibility of a long delay between an action and the resulting reinforcement feedback information.

The FALCON-R is an extension to the actor-critic reinforcement learning or AHC model [9], shown in Fig. 3. It integrates two FALCON-ART networks into a functional unit: the action network is used as a fuzzy controller, and the other, a critic network is used as a fuzzy predictor. They receive the same state input from the environment and go through different inference processes. The action network decides the best action to impose on the environment in the next time step according to the current environment status and the predicted reinforcement signals. The critic network models the environment such that it can make a single or multi-step prediction of the reinforcement signal that will eventually be obtained from the environment for the current action chosen by the action network.

As long as a different fuzzy neural network inference engine has the ability to derive rules from the given sets of patterns, the fuzzy neural network can act as critic or action network. It is not necessarily the case that it must be constituted using FALCON-ARTs. Furthermore, the two functional units (action network and critic network) need not be of the same network structure, although having the two networks the same simplifies the design.

The predicted reinforcement signal can also provide the action network with an estimate of more detailed reward or

Fig. 3. Structure of the FALCON-R network.

penalty information (called internal reinforcement signals) about the candidate action for the action network to learn. The internal reinforcement signal can minimize the uncertainty that the action network faces and speed up the learning process. Moreover, learning can proceed even during the period when no external reinforcement feedback is available.

Associated with the FALCON-R structure is a reinforcement structure-parameter learning algorithm composed of the techniques of temporal difference, stochastic exploration, and the on-line supervised structure-parameter learning algorithm of FALCON-ART. The flowchart of this learning algorithm is shown in Fig. 4. For each input vector from the environment, starting at the input nodes, a forward pass computes the activity levels of all the nodes in the network. Stochastic exploration takes place at the output node of the action network to predict its output error, and temporal difference prediction takes place at the output node of the critic network to predict its output error. When the output error information is available, the reinforcement learning problem is subsequently transformed to a supervised learning problem and then the on-line supervised learning algorithm of the FALCON-ART can be applied directly.

3.2.1. The multi-step critic network

In the case of temporal difference methods, by letting $\lambda = 0$ for efficiency and accuracy [1], a general learning rule used for infinite discounted predictions is given by Eq. (3) as a special case of Eq. (1).

$$\Delta w_t = \eta(r_t + \xi p_t - p_{t-1}) \nabla_w p_{t-1}, \quad (3)$$

Fig. 4. Reinforcement structure/parameter learning algorithm of FALCON-R.

where

- w_t , is the critic network's current parameter,
- x_t , is the current input state vector at time step t ,
- p_t , is the single predicted output of the network, and
- $r_t \in \{-1, 0\}$, is the external reinforcement signal at time step t and is caused by the inputs and actions chosen at earlier time steps (i.e., at time steps $t-1, t-2, \dots$), such that $r_t = 0$ means "success" or during the period when there is no external reinforcement signal, when $r_t = -1$ means "failure".

The infinitely discounted prediction is used for prediction problems in which exact success or failure may never be completely known. Let r'_t be the multi-step critic network's output node error signal (i.e., the internal reinforcement signal),

$$r'_t = r_t + \xi p_t - p_{t-1}, \quad 0 \leq \xi < 1. \quad (4)$$

With the predicted output, p_t , and the output error signal, r'_t , available, the critic network can be trained using the supervised learning techniques of FALCON-ART. If the envi-

ronment provides a reinforcement signal at each time step, the critic network can "single-step", or calculate the actual learning output error at each time step. Thus, the critic network can operate as either a multi-step or single-step predictor.

Given the input training data at time t , $x_i(t)$, $i=1, \dots, n$, the FALCON-ART learning procedure of the critic network consists of two steps: the structure learning step using the fast-learn fuzzy ART with desired output value as the external reinforcement signal, $r(t)$, and parameter learning using the backpropagation algorithm with error function

$$E = \frac{1}{2}(r_t + \xi p_t - p_{t-1})^2. \quad (5)$$

Gradient information is then derived as

$$\left[\frac{\partial E}{\partial p} \right]_{t-1} = p_{t-1} - r_t - \xi p_t = -r'_t. \quad (6)$$

3.2.2. The action network

The reinforcement structure/parameter learning for the action network is to adjust the parameters (e.g., w) of the action network, to change the connection types, or even to add new nodes, such that the reinforcement signal is maximal; that is

$$\Delta w \propto \frac{\delta r}{\delta w}. \quad (7)$$

To determine $\delta r / \delta w$, it is necessary to know $\delta r / \delta y$, where y is the output of the action network. Since the reinforcement signal does not provide any hint as to what the right answer should be in terms of a cost function, there is no gradient information, and the gradient, $\delta r / \delta y$, can only be estimated. After estimating, the online supervised structure/parameter learning algorithm can be directly applied to the action network to solve the reinforcement learning problem.

To estimate the gradient information in a reinforcement learning network, there needs to be some randomness in how the output actions are chosen by the action network so that the range of possible outputs can be explored to determine a correct value. Thus, the output nodes of the action network are designed to be stochastic units, which compute their output as a stochastic function of their input.

The gradient information, $\delta r / \delta y$, is also estimated by the stochastic exploration method [15,16]. In estimating the gradient information, the output y of the action network does not directly act on the environment. Instead, it is treated as a mean (expected) action. The actual action, y' , is chosen by exploring a range around this mean point. This range of exploration corresponds to the variance of a probability function and is the normal distribution.

The stochastic exploration, $\sigma(t)$, is some nonnegative, monotonically decreasing functions of the predicted

reinforcement signal.

$$\sigma(t) = \frac{k}{2} [1 - \tanh(p(t))] = \frac{k}{1 + e^{2p(t)}}, \quad (8)$$

where k is a real number in the range $[0,1]$.

The $\sigma(t)$ can be interpreted as the extent to which the output node searches for a better action. Since $p(t)$ is the predicted reinforcement signal, if $p(t)$ is small, the exploratory range, $\sigma(t)$ will be large. Conversely, if $p(t)$ is large, $\sigma(t)$ will be small. This amounts to narrowing the search about the mean, $y(t)$, if the predicted reinforcement signal is large. This can provide a higher probability of choosing an actual action, $y'(t)$, which is very close to $y(t)$, since it is expected that the mean action $y(t)$ is very close to the best action possible for the current given input vector. On the other hand, the search range about the mean, $y(t)$ is broadened if the predicted reinforcement signal is small such that the actual action has a higher probability of being quite different from the mean action, $y(t)$.

A predicted reinforcement signal is necessary to determine the search range, $\sigma(t)$. This predicted reinforcement signal can be obtained from the critic network. Once the variance has been decided, the actual output of the stochastic node can be set as

$$y'(t) = N(y(t), \sigma(t)), \quad (9)$$

where $y'(t)$ is a normal or Gaussian random variable with density function

$$\text{Prob}(y') = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y'-y)^2}{2\sigma^2}}. \quad (10)$$

4. Performance evaluation of the FALCON-R classifier

The FALCON-R classifier was evaluated with 2 sets of data, Fisher's Iris classification [12], and the X-OR dilemma. During the evaluation, FALCON-R, implementing a reinforcement learning method, was compared against a basic fuzzy neural network, FALCON-ART. The aim of this evaluation is neither to prove or disprove that a reinforcement learning method outperforms conventional supervised learning in a classification problem, but to identify some strengths of reinforcement learning in classification applications, and to provide guidelines to such a formulation.

4.1. Fisher's Iris classification

Fisher's Iris data were first used by Ronald Fisher in an article [17] motivating the statistical technique of discriminant analysis. There are 4 discriminating variables—petal and sepal length and petal and sepal width—and 3 species of iris (Setosa, Virginica and Versicolor). It consists of 50 specimens for each of the species. The features are as shown in Table 1.

Table 1
IRIS data

Attribute	Range in mm
Sepal length	[43,79]
Sepal width	[22,44]
Petal length	[10,69]
Petal width	[1,25]

The performance of the FALCON-R network was evaluated based on the classification performance test of Fisher's Iris data set. The Fisher's Iris data was broken into 2 groups, one group was used for training of the network (training group) and the other group was used to test the performance of the network (control group). These two groups are mutually disjoint; with the training group comprising 35% (50 sets) and the control group containing 65% (99 sets) of the total data.

In the Iris data classification test, there exist sets of correct "target" output values or classification. The classification is the species of the Iris, whether it belongs to Setosa, Virginica or Versicolor. The action network should be able to correctly distinguish the three different classes. The actual action should be able to have physical meaning of showing what the classification result is, and should be able to maximize the reinforcement signal at the next training cycle.

Having the above limitation, the mean action, $y(t)$ of the action network is chosen to be the actual classification output of iris classification in a supervised manner. For the action network, there are three possible outputs: it will fire an action (showing 1's) and deactivate the other output (showing 0's) for the classification it belongs to. The actual action, $y'(t)$ is simply derived using the networks current parameters (e.g., weight, w) instead of using a purely stochastic exploration method.

From the above discussion, the gradient information is estimated as

$$\frac{\delta r}{\delta y} \approx r'(t) \left[\frac{y'(t-1) - y(t-1)}{\sigma(t-1)} \right] \equiv r'(t) \left[\frac{y' - y}{\sigma} \right]_{t-1}, \quad (11)$$

where the subscript, $t-1$, represents time displacement.

If $r'(t) > 0$, the actual action, $y'(t-1)$, is better than the expected action, $y(t-1)$. Then $y(t-1)$ should be moved closer to $y'(t-1)$. Alternatively, if $r'(t) < 0$, the actual action $y'(t-1)$ is worse than the expected action $y(t-1)$ the $y(t-1)$ should be moved further away from $y'(t-1)$.

The desired output values can be estimated as

$$y_d(t) \approx y(t) + k \frac{\delta r}{\delta y} \equiv y(t) + kr'(t) \left[\frac{y' - y}{\sigma} \right]_{t-1}, \quad (12)$$

where k is a real number in the range $[0,1]$.

An action network parameter learning step can be performed in exactly the same way for the critic network parameter learning except that the error term, r'_i is replaced by $\delta r/\delta y$, which is the gradient information for the action network.

For the performance evaluation, the following parameters were used:

4.1.1. For the FALCON-R network

- Discount-rate parameter, $\xi = 0.2$,
- k value for desired output formulation of action, $k = 0.5$,
- cutoff point for formulation of external reinforcement signal = 0.1 (10%),
- training group = 35% (50 data sets),
- control group = 65% (99 data sets),
- percentage of reinforcement given initially = 100%.

Note that in the later stage of performance evaluation, the percentage of reinforcement given was gradually reduced from 100% to 0% to test the multi-step prediction ability performance of FALCON-R for the critic network.

The discount-rate parameter determines the extent of short-range or long-range prediction, so it is closely related to the percentage of reinforcement signal given in the training. In other words, as the percentage of reinforcement signal is gradually reduced, the range of prediction should be gradually increased. But for experimental simplicity, the discount rate parameter was fixed at a constant value for different multi-step performance assessment.

4.1.2. For the action and critic networks

- Learning constant in the backpropagation algorithm, $\eta = 0.005$,
- input vigilance parameter for the fuzzy ART algorithm, $\rho_{in} = 0.8$,
- output vigilance parameter for the fuzzy ART algorithm, $\rho_{out} = 0.8$,
- sensitivity parameter of the trapezoidal membership function, $\gamma = 30$,
- targeted total error = 0.00005.

Note that n different sets of parameters can be used for the action and critic networks. The reason that the same parameter sets were being used for both action and critic network was because the input data sets were the same, and the nature of both networks were assumed to be similar.

To measure the convergence of the FALCON-R network, the error functions for action network and critic network are as expressed in Eqs. (13) and (14).

$$E_{critic} = \frac{1}{2} \sum_{i=1}^n (r_t + \xi p_t - p_{t-1})_i^2 = \frac{1}{2} \sum_{i=1}^n (r'_i)_i^2, \quad (13)$$

$$E_{action} = \frac{1}{2} \sum_{i=1}^n \left(\frac{\delta r}{\delta y} \right)_i \approx \frac{1}{2} \sum_{i=1}^n \left(r'_i(t) \left[\frac{y' - y}{\sigma} \right]_{t-1} \right)_i, \quad (14)$$

where n is the number of outputs for the action or critic network.

The critic and action error functions E were computed for both networks for each training data set and the total error (TE) was computed using Eq. (15) for one training epoch as the summation of all the error functions in that epoch. Each action network and critic network has its own total error based on their error function.

$$TE = \sum_{t=1}^T E_t, \quad (15)$$

where t is the time step of the training data sets (the numbering of training data sets) in one training epoch.

The performance of FALCON-R was evaluated in different aspects as:

1. Two types of reinforcement signal were used, a single reinforcement signal in the critic network based on the classification result or multiple reinforcement signals in the critic network based on a number of outputs of the action network.
2. Different levels of multi-step critic network prediction, ranging from 100% prediction (i.e., single-step prediction) to 0% prediction (no reinforcement signal provided).
3. Tolerance of the FALCON-R network to disturbance added during the training process.

In some cases, the FALCON-R network was compared against the performance of the FALCON-ART network. The FALCON-ART network was setup with the same configuration as the action network or critic network.

4.1.3. Comparison of single critic output and multiple critic output

These two types of FALCON-R have different external reinforcement signal formulations as discussed in Section 2.2. The first type is the general formulation of the reinforcement signal, where the number of reinforcement signals (i.e., number of critic network outputs) is based on the number of action network outputs. Each reinforcement signal is responsible for the prediction of its corresponding action at the action network. This is the multiple critic output case of FALCON-R. The second type is the formulation of the external reinforcement signal based on the classification test. For this case, only a single reinforcement signal needs to be predicted, and the single reinforcement signal is derived from the classification result of the current input data sets. The single reinforcement signal is responsible for

Fig. 5. A plot of total error (TE) against the number of training iterations.

criticizing all the outputs of the action network. This is the single critic output case of FALCON-R.

For the Iris classification test, the critic output was 3 for the multiple critic output case, and 1 in the single critic. Firstly, the total error of the action network for these two types of network was examined. The result of these two types of FALCON-R network was compared against a FALCON-ART network using the same training and testing data set.

Fig. 5 shows the total error versus training iterations for these three networks. The calculation of Total Error for the two types of FALCON-R was different from FALCON-ART, but the convergence gives a sense of the relative behavior of FALCON-R and FALCON-ART.

The total error remained unchanged for the FALCON-ART network, because it could not successfully classify the 2nd and 3rd classes of Iris data. For the case of FALCON-R with multiple critic output, the action output error gradually decreased as expected. This was because the reinforcement signals were gradually being predicted to a range of acceptable error, causing the output of the action to be in a range under controllable conditions. For the case of the single critic output of FALCON-R, the total error did not converge. This was because the 2nd and 3rd classes of the Iris set could not be correctly classified to an acceptable level, and hence negative reinforcement was provided whenever a prediction failure, or misclassification occurred. This caused the network to explore a wider range of possibilities, and the action network total error to become oscillatory. But we can see that at the earlier stage of the training epoch, the single critic output FALCON-R can converge faster than the multiple critic output FALCON-R.

The classification result was examined for the three kinds of network formulation. The results can be seen in Fig. 6. The FALCON-ART network could not successfully classify the 2nd and 3rd classes of Iris data, so the number of successfully classified classes is about $\frac{2}{3}$ of the total testing set—99 items. A similar situation occurred initially with the two types of FALCON-R networks, but as the training progressed, negative reinforcement is given to the misclassified data, causing the network to explore further. Thus, the FALCON-R classification result is oscillatory. At the earlier

Fig. 6. A plot of classification result of successfully classified data.

stage of training, the single critic FALCON-R performed better than the multiple critic FALCON-R. As the training progressed, the multiple critic FALCON-R saturated at a stage where the classification result could not be improved further. This was because the predicted reinforcement signal was already limited to a small range. The single critic FALCON-R can explore wider and faster causing the classification result to remain oscillatory.

The FALCON-ART and the two FALCON-R networks all failed to converge. The cause of this problem can be explained by examining the fuzzy sets generated by the action network for a single critic FALCON-R after five training epochs as shown in Fig. 7. The Virginica and Versicolor classes are much alike from the overlapping of their respective fuzzy sets. There is only a slight difference between the two classes. The total error and the classification result are oscillatory because of the negative reinforcement given to the network when misclassification occurs. This causes the action network to explore further thus leading to oscillation.

The number of fuzzy rules derived from the networks after training was repeated for 500 training iterations is shown in Table 2. The fewer the number of derived fuzzy rules, means that the mapping process of the fuzzy ART algorithm is performing well to map the input hyperboxes to output hyperboxes. In the case where the fuzzy rules are hard to derive (i.e., a complicated system) or the network is out of control, the number of fuzzy rules will increase to a large number and continue growing because it cannot find accurate fuzzy rules to predict the system. The drawback of deriving more fuzzy rules is that the network becomes too big and significantly slows down the training process.

From the above discussion and observation of Table 2, we can conclude that the single critic FALCON-R performed better than the multiple critic FALCON-R, in terms of convergence speed and the number of iterations taken to derive the fuzzy rules to predict the reinforcement signal.

Fig. 7. Fuzzy sets derived from single critic FALCON-Rs action network after five training epochs.

Table 2
Number of fuzzy rules and output derived for different networks

Network Type->	Single critic FALCON-R		Multiple critic FALCON-R	
	Critic	Action	Critic	Action
No. of Fuzzy rules	17	9	46	6
No. of output	1	3	3	3

4.1.4. Effect of multi-step critic FALCON-R

In this analysis, a single output multi-step critic FALCON-R network was used. Different levels of multi-step critic networks were formulated for performance evaluation. These ranged from 100% (single-step critic) to 0% (no reinforcement signal given). The formulation of the multi-step FALCON-R network was achieved by varying the percentage of reinforcement signal given to the FALCON-R network for training. When no reinforcement signal is given, the external reinforcement signal is zero.

The performance of FALCON-R under various levels of reinforcement signal was examined and compared against each other to evaluate the performance of using the temporal difference method, which was originally applied in control applications, and its applicability to classification applications.

The multi-step FALCON-R network was formulated as follows:

IF (a randomized number within the range 0–100 > percentage),
THEN no reinforcement signal is given,
ELSE formulate an external reinforcement signal,

where ‘percentage’ determines the percentage of external reinforcement signal given for the system.

This is derived from the real-world situation where the system is interacting with the environment. The environment will supply an external reinforcement signal in a random number of time steps. At other times, no reinforcement signal will be given and the system needs to set the external

Fig. 8. Total error of the multi-step FALCON-R network with different percentages of reinforcement signal.

reinforcement signal to zero when it does not receive any reinforcement signal.

By formulating the system in this way, the multi-step FALCON-R can be evaluated while at the same time proving that the general case of temporal difference equation (i.e., prediction of infinite discounted cumulative outcomes) is workable for formulating a FALCON-R network as a classifier.

The total error and classification result for ten training iterations of FALCON-R with different percentages of reinforcement signal is shown in Fig. 8.

As can be seen from Fig. 8, the performance of the multi-step FALCON-R is acceptable from 100% prediction (single-step prediction) to 50% prediction. The performance degrades to an unacceptable situation at 25% prediction, while at 0% prediction (no prediction signal is given), the network performance is similar to a basic FALCON-ART network. This demonstrates that the system is workable as a classifier when an acceptable amount of reinforcement signal (not necessarily 100% reinforcement) is given during the training process. With the existence of the external reinforcement signal, the FALCON-R network performance is superior to a normal FALCON-ART classifier. As discussed previously, in the real world, an external reinforcement signal is hard or expensive to obtain. With the above proven characteristics of FALCON-R, the number of external reinforcement signals given to the system can be reduced to a certain extent thus saving the cost of deriving such external reinforcement signals in real world environments.

In the original formulation of FALCON-R as a control system, the temporal difference method was used so that multi-step prediction ability is possible. In formulating FALCON-R as a classification system, multi-step prediction is not realizable since the classification data are not temporally related to each other. But the classification system is still workable when multi-step external predictions are given. In the formulation of this classification system, negative reinforcement was set at -1 , positive reinforcement was set at $+1$, and when no external reinforcement was given, the reinforcement signal was set at 0 . In this situation, there will be no effect when a positive reinforcement signal is applied. When a negative reinforcement signal of

Fig. 9. Classification result of Multi-step FALCON-R with different percentages of reinforcement signal.

-1 is applied, the system will be in a failure state and will explore for a better state. The system is then attempting to control the false state to an acceptable positive situation, where classification performance is maximized. In this situation, multi-step formulation for the FALCON-R classifier is still possible.

Fig. 9 shows the classification result on the set of 99 test data using FALCON-R with different levels of reinforcement signal. The numbers of correctly classified patterns are shown with different levels of reinforcement signal for the critic network. As in the case for total error, the performance was significantly better than the normal FALCON-ART network when a reinforcement signal between 50% and 100% was used. As can be seen, more than 90% of the data are correctly classified starting from the second training iteration. The classification performance degraded when less than 25% reinforcement signal was given. When 0% reinforcement signal was applied (no external reinforcement signal given to the critic network), the result is similar to the normal FALCON-ART network.

One of the major differences between FALCON-ART and FALCON-R is that, in FALCON-R training, the total error does not necessarily reduce gradually with a very smooth transition as in the FALCON-ART network. Once there is misclassified data, the network will attempt to explore further and thus cause the total error to be very much different from the previous iteration when the network explores in the wrong direction. As a classifier, as long as a suitable reinforcement signal is applied, the overall performance of FALCON-R is better than the normal FALCON-ART network.

Table 3 summarizes the network behavior of FALCON-R with different percentages of external reinforcement given for multi-step prediction. We can see that the critic and action networks are able to derive correct fuzzy rules when 50–100% of reinforcement is given. When 25% of reinforcement is given, the network is still able to derive fuzzy rules from the training data, but the performance (in terms of numbers of successfully classified data) is already degraded. When no reinforcement signal is applied, the critic

Table 3
Number of fuzzy rules derived in the FALCON-R network after 10 iterations (12 iterations for 0% reinforcement)

% of reinforcement	Critic	Action
100	17	3
75	17	3
50	17	3
25	35	5
0	612	3

Fig. 10. FALCON-ART training with disturbance added at epoch number five.

network is unable to derive correct fuzzy rules. This causes the critic networks to grow very fast. In just 12 iterations, the critic network grew to 612 fuzzy rules. In each cycle, an additional fuzzy rule is assigned to the new training pattern, thus making the number of fuzzy rules to be 612 after 12 iterations with 51 sets of training data for each iteration. The classification result for this case (0% external reinforcement given) is similar to FALCON-ART classification, but due to the fact that the critic grows too large, the training speed degrades to an unacceptable extent.

4.1.5. Disturbance rejection capability

In order to demonstrate the disturbance rejection capability of the FALCON-R system, a disturbance was introduced to the system during the training process. One of the data sets from class 1 was intentionally wrongly set as class 3 during the fifth training iteration. The behavior of FALCON-R under disturbance is compared against a normal FALCON-ART network in Figs. 10 and 11. As can be seen, the total error after disturbance does not return to the normal condition when no distortion is added. This shows that the disturbance rejection ability of FALCON-R is superior to that of a FALCON-ART classifier.

During disturbance conditions, the network wrongly classified the training data as a new pattern. However, after a few training iterations, due to the effect of reinforcement and critic given to the system, the disturbance is eliminated

Fig. 11. FALCON-R training with disturbance added at epoch number five.

Fig. 12. The X-OR dilemma.

Table 4
Parameters used for X-OR dilemma

FALCON-R	Critic & action
$\xi = 0.2$	$\eta = 0.005$
$k = 0.5$	$\rho_{in} = 0.8$
Cutoff = 0.1 (10%)	$\rho_{out} = 0.8$
% reinforcement = 100%	$\gamma = 30$

by the system. This is one of the major advantages of a fuzzy neural network classifier with reinforcement learning ability.

4.2. X-OR dilemma

The classification ability of FALCON-R was further evaluated with the X-OR dilemma. The X-OR dilemma involves learning and classifying the four corners of a unit square in 2-D space, as shown in Fig. 12.

All four corners will be used in training, while the output will be either class 0 or class 1. The network parameters used for this simulation are shown in Table 4.

In this simulation, only the multiple-critic (2-critic) FALCON-R network with 100% reinforcement was used. The network learns to correctly classify the 4 corners in a single training iteration, with 100% classification rate.

Four fuzzy rules were derived as follows:

1. If both inputs are zeros, then output is Class 0.
2. If both inputs are ones, then output is Class 0.
3. If first input is one and second input is zero, then output is Class 1.
4. If first input is zero and second input is one, then output is Class 1.

As shown by Kim [18], the training epoch for a conventional back-propagation network is 320 iterations. The FALCON-R outperforms a conventional back-propagation neural network because of the fuzzy ART-like clustering technique being used, which is suitable for a classification application, compared to the conventional back-propagation network, which is suitable for training of regressive data.

4.3. Strengths for applying reinforcement learning to a classifier

From the above discussion, we can conclude that a FALCON-R based classifier possesses the following advantages when compared to the FALCON-ART classifier system.

1. The FALCON-R is able to easily escape from local minima during the training process. This happens whenever the system receives a negative reinforcement signal from the environment, causing the system to explore further to seek a potentially better state by getting out of the local minima.
2. As long as the reinforcement signal formulation is appropriate, the training iteration of FALCON-R is faster than a normal FALCON-ART classifier. Note that one training cycle of FALCON-R involves a training cycle of action network and a critic network. The system adapts to the environment faster due to the existence of the applied reinforcement signal.
3. FALCON-R disturbance rejection ability is excellent as it can quickly recover to a stable state after a disturbance is added. But for a FALCON-ART network, the system can hardly tolerate such a disturbance.

5. Conclusions and recommendations

Reinforcement learning systems have been applied successfully in the area of Game Playing and Control applications. This paper discusses the application of reinforcement learning in classification or model matching. FALCON-R, which was originally a reinforcement learning neural-fuzzy controller, has been formulated as a classifier for solving classification problems. By combining temporal difference techniques, stochastic exploration and an online supervised structure/parameter learning algorithm a

reinforcement structure/parameter learning algorithm was formulated for the FALCON-R classifier. Using the formulated FALCON-R classifier, the classification of Fisher's Iris data was attempted and the performance was compared against a normal FALCON-ART classification system. The classifier was further evaluated with the X-OR problem. In the formulation of the classification system, the basic FALCON-R network was modified appropriately to suit the classification application.

Comparing it against the result of a FALCON-ART network identifies some strengths of using a FALCON-R network as a classifier. For example, the FALCON-R training speed is much faster than FALCON-ART due to the prediction ability of the reinforcement signal in the FALCON-R network. The learning ability in a stochastic environment of FALCON-R is much better than FALCON-ART. The FALCON-R network is able to tolerate disturbance during the training process, while the FALCON-ART network responded poorly to the disturbance test.

In the future, investigation of reinforcement learning can be extended to produce a more suitable algorithm for application in classification systems. While the function approximator can be a neural fuzzy system other than the FALCON-ART network, the performance may be different from FALCON-R, which depends on the abilities of the other fuzzy neural networks to derive the critic needed (for critic) or suitable action (for action network). In this paper, only the derivation of reinforcement learning using FALCON-ART networks (FALCON-R) is discussed. The application of reinforcement learning continues to be a research issue and could be extended to application areas such as forecasting or prediction.

References

- [1] R.S. Sutton, Learning to predict by the methods of temporal differences, *Mach. Learning* 3 (1998) 9–44.
- [2] D.P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 1995.
- [3] J.H. Holmes, D.R. Durbin, F.K. Winston, The Learning classifier system: an evolutionary computation approach to knowledge discovery in epidemiologic surveillance, *Artif. Intell. Med.* 19 (1) (2000) 53–74.
- [4] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, 1998.
- [5] J. Peng, B. Bhanu, Closed-loop object recognition using reinforcement learning, *IEEE Trans. Pattern Anal. Mach. Intell.* 20 (2) (1998) 139–154.
- [6] J. Peng, B. Bhanu, Delayed reinforcement learning for adaptive image segmentation and feature extraction, *IEEE Trans. Syst. Man Cybern.-Part C: Appl. Rev.* 28 (3) (1998) 482–488.
- [7] R. Greiner, A.J. Grove, D. Roth, Learning cost-sensitive active classifiers, *Artif. Intell.* 139 (2) (2002) 137–174.

- [8] C.T. Lin, C.J. Lin, Reinforcement learning for an ART-based fuzzy adaptive learning control network, *IEEE Trans. Neural Networks* 7 (3) (1996) 709–731.
- [9] C.T. Lin, C.S.G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice-Hall, Englewood cliffs, 1996.
- [10] C.J. Lin, C.T. Lin, An ART-based fuzzy adaptive learning control network, *IEEE Trans. Fuzzy Systems* 5 (4) (1997) 477–496.
- [11] R.S. Sutton, Temporal credit assignment in reinforcement learning, Ph.D. Dissertation, University of Massachusetts, Amherst, MA, 1984.
- [12] D.P. Bertsekas, J.N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
- [13] J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, New York, 1991.
- [14] G.A. Carpenter, S. Grossberg, D.B. Rosen, Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system, *Neural Networks* 4 (6) (1991) 759–771.
- [15] J.A. Franklin, Input space representation for reinforcement learning control, in: *Proceedings of the IEEE International Symposium on Intelligence Control*, 1989, pp. 115–122.
- [16] C.T. Lin, C.S.G. Lee, Reinforcement structure/parameter learning for neural network based fuzzy logic control systems, *IEEE Trans. Fuzzy Systems* 2 (1) (1994) 46–63.
- [17] R.A. Fisher, The use of multiple measurements in taxonomic problems, *Ann. Eugenics* 7 (1936) 179–188.
- [18] K.B. Kim, M.H. Kang, E.Y. Cha, A fuzzy self-organized backpropagation using nervous system, *IEEE International Conference on Systems Man and Cybernetics*, vol. 2, 1997, pp. 1457–1462.

FITSK: Online Local Learning With Generic Fuzzy Input Takagi–Sugeno–Kang Fuzzy Framework for Nonlinear System Estimation

Kian Hong Quah and Chai Quek, *Member, IEEE*

Abstract—Existing Takagi–Sugeno–Kang (TSK) fuzzy models proposed in the literature attempt to optimize the global learning accuracy as well as to maintain the interpretability of the local models. Most of the proposed methods suffer from the use of offline learning algorithms to globally optimize this multi-criteria problem. Despite the ability to reach an optimal solution in terms of accuracy and interpretability, these offline methods are not suitably applicable to learning in adaptive or incremental systems. Furthermore, most of the learning methods in TSK-model are susceptible to the limitation of the curse-of-dimensionality. This paper attempts to study the criteria in the design of TSK-models. They are: 1) the interpretability of the local model; 2) the global accuracy; and 3) the system dimensionality issues. A generic framework is proposed to handle the different scenarios in this design problem. The framework is termed the generic fuzzy input Takagi–Sugeno–Kang fuzzy framework (FITSK). The FITSK framework is extensible to both the zero-order and the first-order FITSK models. A zero-order FITSK model is suitable for the learning of adaptive system, and the bias-variance of the system can be easily controlled through the degree of localization. On the other hand, a first-order FITSK model is able to achieve higher learning accuracy for nonlinear system estimation. A localized version of recursive least-squares algorithm is proposed for the parameter tuning of the first-order FITSK model. The local recursive least-squares is able to achieve a balance between interpretability and learning accuracy of a system, and possesses greater immunity to the curse-of-dimensionality. The learning algorithms for the FITSK models are online, and are readily applicable to adaptive system with fast convergence speed. Finally, a proposed guideline is discussed to handle the model selection of different FITSK models to tackle the multi-criteria design problem of applying the TSK-model. Extensive simulations were conducted using the proposed FITSK models and their learning algorithms; their performances are encouraging when benchmarked against other popular fuzzy systems.

Index Terms—Degree of localization, localized learning, nonlinear system estimation, Takagi–Sugeno–Kang fuzzy models, zero and first-order TSK models.

I. INTRODUCTION

THE Takagi–Sugeno–Kang (TSK) fuzzy model [1]–[3] is a class of fuzzy models that assumes local model representations with local function dynamics at the consequent or rule-layer of the models. The idea of such a fuzzy model is to consider the output by performing fuzzy interpolations of

these simpler local functional models in the neighboring fuzzy partitions. The main advantage of the TSK-model over other classes of fuzzy models is its ability to model a system accurately; either globally or locally. The accurate global learning ability motivates the practical applications of TSK-model in nonlinear system estimation [4]. The local learning ability provides a course of interpretability of the local models in the localized subspaces [5]–[8].

Existing TSK-model learning algorithms proposed in the literature can be classified using two criteria. The first criterion is essentially the locality of learning. The second criterion is being an online or an offline learning algorithm. The locality of learning depends on the model's learning objective function, which is a minimization problem of the global or the local learning errors. The global parameter tuning considers the minimization of the global error of the model, such as the ANFIS's Kalman filter algorithm [9]. This benefits the global accuracy of the system, but suffers from degradation in local interpretability. This degradation exhibits an erratic local behavior, which causes the local models to have difficulty in interpretability [5]. In contrast, the local parameter tuning takes the form of the minimization problem of the local learning errors. This ensures that the local models possess accurate representations and improves on the local interpretability of the system. Examples of such local parameter tuning are the direct update of zero-order TSK model and the local learning algorithm in Yen's model [5]. However, minimization of local learning errors without consideration of global behavior results in degradation of global learning accuracy.

An offline learning algorithm assumes that the data is presented in a batch form and can be repeatedly accessed. This form of learning is easy to guarantee its success in reaching an optimal solution based on its learning objective function. Furthermore, it has the flexibility in recalling the stored training examples to improve the quality of learning. However, the offline learning is inapplicable to adaptive or incremental learning systems. Moreover, it is prone to the *curse-of-dimensionality* problem [10]. Examples of the offline learning methods are the singular value decomposition approach [5] and the constraint-optimization method [8]. Summarizing these, the existing TSK-models encountered one or more of the following major problems despite being good modeling tools for nonlinear system estimation. They are; namely: 1) offline learning; 2) low interpretability of local model; 3) degradation in global learning accuracy; and 4) inapplicable to higher dimensionality system.

Manuscript received November 18, 2004; revised May 23, 2005. This paper was recommended by Associate Editor Hideyuki Takagi.

The authors are with the Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: ashcquek@ntu.edu.sg).

Digital Object Identifier 10.1109/TSMCB.2005.856715

Hence, a novel TSK-model framework that is immune to the above deficiencies is proposed in this paper. The framework is termed the *generic fuzzy input Takagi-Sugeno-Kang fuzzy framework* (FITSK). The FITSK framework can be readily extended to different classes of TSK-models to alleviate the above deficiencies based on different design criteria. All the learning algorithms of FITSK assume online learning paradigms with fast convergence speed.

This paper is organized as follows. Section II describes the general structure of the FITSK framework, the zero-order and the first-order FITSK models, and their online learning algorithms. Section III presents the simulation results of the FITSK models with five different nonlinear system estimation tasks. Section IV concludes this paper.

II. FITSK

This section introduces the generic FITSK framework; the FITSK framework takes the form of a superset of all the TSK fuzzy models [1]–[3]. A special case of the FITSK framework is mapped to a zero-order FITSK model (denoted by FITSK^0) on the assumption that the local models' function dynamics are zero-order (constant term). Subsequently, a degree of localization of the zero-order FITSK model is introduced to balance the bias-variance of the learning of the FITSK model. A first-order FITSK model (denoted by FITSK^1) is another adaptation of the FITSK framework, where the local models are first-order linear functions of the input vector. There are two classes of parameter learning for the FITSK^1 model. They are the global recursive least-squares algorithm and the local recursive least-squares algorithm. The latter is computationally more effective than the former. A degree of localization is introduced in the local recursive least-squares algorithm to control the interpretability of the FITSK^1 model. A guideline is proposed to select the choice of different FITSK models with respect to the following three design criteria; namely 1) model interpretability; 2) learning accuracy; and 3) data dimensionality. The guideline allows the choice of the FITSK models with different settings to generate a balanced system on the basis of these criteria.

A. Generic Structure of FITSK

Fig. 1 depicts the generic structure of FITSK. The FITSK framework consists of six layers of nodes. The six layers are the input fuzzifier (IF), the input linguistic (IL), the conjunction (C), the normalization (N), the functional or rule (F), and the summation (S) layers. The input fuzzifier nodes in the first layer fuzzify the input vector into a corresponding vector of fuzzy membership functions. Subsequently, the input linguistic layer measures the similarity of each fuzzy membership functions with its corresponding linguistic nodes. The conjunction layer computes the conjunction of all incoming signals and generates the corresponding weights. The normalization layer normalizes these weights. Each node of the functional or rule layer implements a local model, which is a form of the TSK rule. The outputs of the functional layer are weighted with their incoming normalized weights. Finally, the summation layer summarizes all the incoming signals. Therefore, the outputs of the FITSK are computed by interpolating the local models with the normalized

fuzzy measures of the input vector. This paragraph briefly summarizes the operations of the generic FITSK framework; subsequent segment of this section formularizes these mathematically.

Each input fuzzifier node IF_i , $i \in \{1, 2, \dots, I\}$, has a single input. Vector $X = [x_1, \dots, x_i, \dots, x_I]^T$ represents the inputs to the FITSK. Each output node S_n , where $n \in \{1, 2, \dots, N\}$ computes a single output denoted by y_n . Vector $Y = [y_1, \dots, y_n, \dots, y_N]^T$ denotes the outputs of the FITSK with respect to the input vector X . The variables i, j, k, l, m , and n are used to refer to arbitrary nodes in layers 1, 2, 3, 4, 5, and 6, respectively.

The total number of nodes for layers 1, 3, 4, and 6 are denoted with variables I, K, L , and N respectively. Each input fuzzifier node IF_i may have different number of input linguistic (IL) nodes J_i . Hence, the total number of nodes for layer 2 is $\sum_{i=1}^I J_i$. Layer 3 consists of conjunction nodes C_k , where $k \in \{1, 2, \dots, K\}$. Each layer 3 node C_k is directly connected to a layer 4 normalization node N_l . Hence, the total number of nodes in layers 3 and 4 are identical, $K = L$. Each layer 4 node N_l is associated with M number of layer 5 functional nodes, $\text{F}_{l,m}$. M is identical to the number of layer 6 nodes, $M = N$. Hence, the total number of nodes in layer 5 is $L \times M$. Each output node at layer 6 is a summation node, S_n . Each summation node is connected with L number of layer 5 functional nodes. The total number of nodes at layer 6, N , depends on the dimension of output vector Y . The FITSK model adopts the TSK's fuzzy model and the trainable parameters are found in layer 5 of the model.

The output of a node is denoted as Z with the subscripts specifying its origin, for example Z_{IF_i} is the output of node IF_i . All the outputs of a layer are propagated to the inputs of the connecting nodes at the next layer with unity link-weight.

The generic operations of the FITSK can be defined as follows.

Layer 1 (Input fuzzifier layer)

$$Z_{\text{IF}_i} = \mu_i(x_i) \quad (2.1)$$

where $\mu_i(\cdot)$ is the fuzzy membership function of input fuzzifier node IF_i

Layer 1 nodes fuzzify a singleton input by adopting the input x_i as the center of the fuzzy membership function, such that the membership function achieves maximum value at x_i and decreases while moving away from the center. However, when a nonsingleton (fuzzy) input is presented, the node simply directs the nonsingleton input as the output of the node.

Layer 2 (Input Linguistic layer)

$$Z_{\text{IL}_{i,j}} = \mu_{i,j}(Z_{\text{IF}_i}) = \mu_{i,j}(\mu_i(x_i)) \quad (2.2)$$

where $\mu_{i,j}(\cdot)$ is a fuzzy linguistic label of node IF_i .

Layer 2 nodes take in the fuzzy membership function of layer 1 outputs as the inputs, and compute a *fuzzy subsethood measure* [4] for each corresponding fuzzy linguistic label. The fuzzy subsethood measure defines the degree that fuzzy set A is a subset of fuzzy set B as

$$S(A, B) = \frac{|A \cap B|}{|A|} \quad (2.3)$$

Fig. 1. Generic structure of FITSK.

If the *min* operator is used for intersection operation, then (2.3) is described in detail in (2.4)

$$S(A, B) = \frac{\int_{x \in U} \min(\mu_A(x), \mu_B(x))}{\int_{x \in U} \mu_A(x)} \quad (2.4)$$

or can be approximated by (2.5)

$$S(A, B) = \frac{\max_{x \in U} (\min(\mu_A(x), \mu_B(x)))}{\max_{x \in U} (\mu_A(x))}. \quad (2.5)$$

Layer 3 (Conjunction layer):

$$Z_{C_k} = \prod_{i=1}^I Z_{IL_{i,j}}. \quad (2.6)$$

Layer 3 nodes compute the conjunction of all the incoming signals using a product or a minimum inference.

Layer 4 (Normalization layer)

$$Z_{N_i} = \frac{Z_{C_k}}{\sum_{k=1}^K Z_{C_k}} = w_i \quad (2.7)$$

where w_i is the normalized weight.

Layer 5 (Functional/Rule layer):

$$Z_{F_{l,m}} = Z_{N_i} f_{F_{l,m}}(X) = w_i f_{F_{l,m}}(X) \quad (2.8)$$

where $f_{F_{l,m}}(X)$ is a local model function of node $F_{l,m}$.

Layer 6 (Summation layer):

$$y_n = Z_{S_n} = \sum_{l=1}^L Z_{F_{l,m}}. \quad (2.9)$$

Two motivations drive the development of the FITSK model. The first is to craft a systematic and generic framework for an arbitrary order TSK model [1]–[3] with arbitrary parameter learning capability. The second is to map the generic model with a nonsingleton (fuzzy) input fuzzifier such that the *degree of localization* of the local model can be controlled with ease. A special case of the generic FITSK is the ANFIS model [9] which considers a singleton input fuzzifier at layer 1, a product inference scheme at layer 3, a first-order polynomial local model function at layer 5, and a single output at layer 6.

B. A Zero-Order FITSK Model

The zero-order FITSK model is a special case of the generic FITSK framework with zero-order polynomial functions (constant terms) at layer 5, and product inference in layer 3. The specific formulation at layer 1 and 2 are described as follows.

Layer 1 (Input fuzzifier layer): The choice of fuzzy membership function is flexible. In this formulation, a Gaussian membership function is chosen

$$Z_{IF_i} = \mu_i(x_i) = e^{-\frac{(x-x_i)^2}{2\sigma_i^2}} \quad (2.10)$$

where

- x spreads over the universe of possible range of x_i ;
- σ_i standard deviation of the Gaussian function.

The standard deviation σ_i reflects the width of the membership function, and is dependant on the accuracy and generalization to be achieved by a specific system. This is a *bias-variance dilemma* problem. Smaller value of the standard deviation (bias) implies higher focusing region for the local model in the local subspace, and directly proportions to higher learning accuracy. Conversely, a larger value (variance) anticipates higher noise or generalization capability in the neighboring subspaces.

Layer 2 (Input linguistic layer): Each linguistic node is a singleton label to simplify the computation of fuzzy subsethood measure. Let $s_{i,j}$ denotes the centre of the singleton of the j^{th} linguistic label of the i^{th} input node, $\mathbf{IL}_{i,j}$, with unity height. The computations involving layer 1 and layer 2 nodes can be simplified as

$$Z_{\mathbf{IL}_{i,j}} = e^{\frac{-(s_{i,j} - x_i)^2}{2\sigma_i^2}}. \quad (2.11)$$

The system designer would have to specify the number of labels in each dimension. The location of $s_{i,j}$ is evenly distributed over the value range of the input dimension i . The resultant resolution r_i is computed as

$$r_i = \frac{\text{range}_i}{\text{no. of labels}_i - 1} \quad (2.12)$$

where range_i is the value range of the input dimension i and no. of labels_i is the total number of labels in dimension i .

Layer 5 (Functional layer)

$$\begin{aligned} f_{\mathbf{F}_{l,m}}(X) &= c_{\mathbf{F}_{l,m}} \\ Z_{\mathbf{F}_{l,m}} &= w_l c_{\mathbf{F}_{l,m}} \end{aligned} \quad (2.13)$$

where $c_{\mathbf{F}_{l,m}} \in \mathcal{R}$ is the zero-order function (constant term) of the local model in the node $\mathbf{F}_{l,m}$.

Let the vector $D = [d_1, \dots, d_n, \dots, d_N]^T$ denote the actual outputs of the FITSK with respect to the input vector X . The training assumes a direct adaptive learning process, which has more plasticity rather than stability. The learning process adapts to new information rapidly by forgoing past information. Thus, the zero-order FITSK model is effective in modeling nonlinear adaptive system such as time-series data or temporally adaptive system. The learning process for an arbitrary zero-order local model is listed as

$$\begin{aligned} \Delta c_{\mathbf{F}_{l,m}}^t &= \alpha w_l (d_n^t - y_n^t) \\ c_{\mathbf{F}_{l,m}}^{t+1} &= c_{\mathbf{F}_{l,m}}^t + \Delta c_{\mathbf{F}_{l,m}}^t \end{aligned} \quad (2.14)$$

where t is the time index and $\alpha \in (0, 1]$ is the positive step size learning constant.

If the stability of the learning system is crucial, the learning constant can be modified to a monotonically decreasing function to tighten the plasticity of the system.

The design of input fuzzifier with singleton linguistic formulation is driven by two motivations. The first is to exclude the

need of online clustering process for the linguistic labels. The second motivation is to ease the control of the *degree of localization* in the neighboring subspaces.

Clustering process for the Mamdani's fuzzy model [11] is an unsupervised learning process by partitioning the input and output spaces into reasonable clusters [12]–[14]. However, unsupervised clustering process for TSK fuzzy model identification is unjustifiable because the quality of the cluster is dependant on the input data as well as the corresponding output dynamics of the system. Several clustering techniques [3], [15] have been proposed for TSK model identification in a supervised learning manner, in which the quality of the output is quantified with respect to the input clusters formation. However, they suffer from adopting a heuristic search approach and are computationally expensive [3], [15]. This demonstrates that the supervised clustering problem for identification of TSK fuzzy model structure is a combinatorial complex problem. As a result, the FITSK model presupposes an even space partitioning of the singleton linguistic labels instead of performing an effective online clustering technique. Normal TSK model maintains the learning accuracy by fuzzy interpolation of neighboring local models. The FITSK is able to maintain a similar fuzzy interpolation by having fuzzy input with singleton linguistic labels, without the need of having a clustering process to structurally define the optimal linguistic labels. This is because the fuzzification is performed prior to the linguistic layer, and the membership functions are well-defined in the sense that the centers of the membership functions represent the exact location of the input vector.

The standard deviation in (2.11) is defined as a function of the input value range in (2.15)

$$\sigma_i = sr^{ld} \times \frac{\text{range}_i}{5} \quad (2.15)$$

where

- $sr \in \mathcal{R}$ shrinking rate and $sr \in (0, 1]$;
- range_i value range of the input dimension i ;
- ld degree of localization.

The maximum of the standard deviation is simply estimated as $\text{range}_i/5$ such that the membership function is able to contain the entire range of the input space. The minimum of the standard deviation is chosen with respect to the concept of ε -completeness. The level ε at the crossover point of neighboring fuzzy membership functions implies a strong belief in the positive sense of the fuzzy control rules, which are associated with the controller. The degree of belief is chosen to be greater than 0.5 [16], [17] to achieve this. Thus, the weight or matching degree of at least one of the label of an input dimension is selected to be greater than 0.5.

Equations (2.16) and (2.17) summarize the derivation of a minimum of the standard deviation, which is derived from (2.11)

$$\begin{aligned} 0.5 &\leq e^{\frac{-(s_{i,j} - x_i)^2}{2\sigma_i^2}} \\ \Rightarrow \sigma_i &\geq 0.8493 \times (s_{i,j} - x_i). \end{aligned} \quad (2.16)$$

Fig. 2. Motorcycle crash dataset.

The boundary condition for $(s_{i,j} - x_i)$ appears when x_i is in the middle of two adjacent labels, which is half of the resolution r_i . This is shown in (2.17)

$$\begin{aligned} (s_{i,j} - x_i) &\geq \frac{r_i}{2} \\ \therefore \sigma_i &\geq 0.8493 \times \frac{r_i}{2} \\ \Rightarrow \sigma_i &\geq 0.4247r_i. \end{aligned} \quad (2.17)$$

Therefore, the standard deviation is set to be at least 0.4247 times of the resolution in that dimension.

Equation (2.15) is able to control the bias-variance of the learning system by varying the degree of localization, ld . This is illustrated with a motorcycle crash dataset from [18]. The dataset consists of a one-dimensional time-series of 133 accelerometer readings in an experiment on the efficacy of crash helmets, as shown in Fig. 2.

Experiment was performed by randomly splitting the dataset into 67 training dataset and 66 testing dataset; setting the shrinking rate, $sr = 0.95$; and the no. of labels to 20. Each set of the results was acquired by setting a degree of localization, ld , and being trained for 200 iterations.

Fig. 3 illustrates that the bias-variance of the learning system by simply modifying the degree of localization, which directly controls the width of the membership function. The training accuracy of the system is proportional to the width of the membership function. However, the testing error increases when the degree of localization is more than 16, representing a scenario of overfitting or overtraining.

Thus, the bias-variance dilemma is a system dependant design issue. The FITSK model simplifies the design process by using the degree of localization to control the bias-variance. The degree of localization defines the locality or the vagueness of the input vector to neighboring subspaces. Higher degree of localization refers to higher accuracy (bias) and less generalization (variance), and vice-versa. The locality of local models can be controlled with ease by varying the degree of localization of the FITSK model without the need of repositioning or redefining the linguistic labels.

A possible usage of varying the degree of localization is to relate it to the progress of training in an adaptive system, such that the degree of localization is gradually increased as learning progresses. This ensures that the generalization capability of the learning system is guaranteed when learning commences, and gradually moves toward higher learning accuracy.

Fig. 3. Training and testing MSEs of FITSK.

C. A First-Order FITSK Model

The FITSK framework takes the form of a first-order FITSK model by adopting first-order local models at layer 5 (functional layer) of the framework. This is achieved by substituting a first-order linear function into (2.8) of the generic FITSK framework

$$\begin{aligned} Z_{F_{l,m}} &= w_l f_{F_{l,m}}(X) \\ f_{F_{l,m}}(X) &= b_{0,(l,m)} + b_{1,(l,m)}x_1 + \dots \\ &\quad + b_{i,(l,m)}x_i + \dots + b_{I,(l,m)}x_I \end{aligned} \quad (2.18)$$

where $B_{(l,m)} = [b_{0,(l,m)}, b_{1,(l,m)}, \dots, b_{i,(l,m)}, \dots, b_{I,(l,m)}]^T$ refers to the parameter vector of an arbitrary node, $F_{l,m}$, in layer 5 of the FITSK. The inference performed by the FITSK model is an interpolation of all the relevant local linear models based on the relevance of input data to the fuzzy subspaces associated with the linear model. The output nodes at layer 6 take the form of (2.19)

$$\begin{aligned} y_n &= \sum_{l=1}^L w_l f_{F_{l,m}}(X) \\ &= \sum_{l=1}^L w_l (b_{0,(l,m)} + b_{1,(l,m)}x_1 + \dots \\ &\quad + b_{i,(l,m)}x_i + \dots + b_{I,(l,m)}x_I). \end{aligned} \quad (2.19)$$

For simplicity of discussion, assume that the FITSK has two inputs x_1 and x_2 , and a single output y , with only two rules, then

$$\begin{aligned} y &= \sum_{l=1}^2 w_l f_{l,1}(x_1, x_2) \\ &= w_1 (b_{0,(1,1)} + b_{1,(1,1)}x_1 + b_{2,(1,1)}x_2) \\ &\quad + w_2 (b_{0,(2,1)} + b_{1,(2,1)}x_1 + b_{2,(2,1)}x_2). \end{aligned} \quad (2.20)$$

Consider that the learning system is presented with P number of training examples, $(X^{(1)}, d^{(1)}), (X^{(2)}, d^{(2)}), \dots, (X^{(p)}, d^{(p)}), \dots, (X^{(P)}, d^{(P)})$, where

- p index;
- $X^{(p)}$ p^{th} input vector, $p \in \{1, 2, \dots, P\}$;
- $d^{(p)}$ p^{th} desired output value.

Equation (2.20) can be easily arranged in a matrix form of

$$AB = C \quad (2.21)$$

or

$$\begin{bmatrix} w_1^{(1)} & w_1^{(1)}x_1^{(1)} & w_1^{(1)}x_2^{(1)} & w_2^{(1)} & w_2^{(1)}x_1^{(1)} & w_2^{(1)}x_2^{(1)} \\ w_1^{(2)} & w_1^{(2)}x_1^{(2)} & w_1^{(2)}x_2^{(2)} & w_2^{(2)} & w_2^{(2)}x_1^{(2)} & w_2^{(2)}x_2^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_1^{(p)} & w_1^{(p)}x_1^{(p)} & w_1^{(p)}x_2^{(p)} & w_2^{(p)} & w_2^{(p)}x_1^{(p)} & w_2^{(p)}x_2^{(p)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_1^{(P)} & w_1^{(P)}x_1^{(P)} & w_1^{(P)}x_2^{(P)} & w_2^{(P)} & w_2^{(P)}x_1^{(P)} & w_2^{(P)}x_2^{(P)} \end{bmatrix} \times \begin{bmatrix} b_{0,(1,1)} \\ b_{1,(1,1)} \\ b_{2,(1,1)} \\ b_{0,(2,1)} \\ b_{1,(2,1)} \\ b_{2,(2,1)} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \vdots \\ d^{(p)} \\ \vdots \\ d^{(P)} \end{bmatrix} \quad (2.22)$$

with dimensions of matrices A , B , and C being $P \times L$, $L \times 1$, and $P \times 1$ respectively, where L is total the number of linear parameters.

Since the number of training examples is usually greater than the number of parameters to be identified, (2.22) is an over-determined problem and there is no exact solution. However, on-line estimation of the solution is possible with the formulation of recursive-least-squares algorithm (a special case of the Kalman filter algorithm) in order to perform iterative linear least-square estimation (LSE), as described in [9] and formally listed in [19].

Let a_p be the p^{th} row vector of matrix A , then B can be iteratively estimated using (2.23)

$$\begin{aligned} B_{p+1} &= B_p + S_{p+1}a_{p+1}^T \left(d^{(p+1)} - a_{p+1}B_p \right) \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{1 + a_{p+1} S_p a_{p+1}^T} \\ p &= 0, 1, \dots, P-1 \end{aligned} \quad (2.23)$$

with initial conditions of $B_0 = 0$ and $S_0 = \gamma I$, where γ is a large positive number and I is the identity matrix of dimension $L \times L$.

The strengths of using the recursive least-squares algorithm for the first-order FITSK parameter identification are its on-line learning capability, fast convergence speed with minimal learning iterations, and optimal global learning performance [4], [7], [20]. However, it encounters two major problems. Firstly, the recursive-least-squares algorithm has high computational and space complexities. For example, a first-order TSK fuzzy model of I -input, L -rules will have an S matrix of dimension $L^2(I+1)^2$, and the computational cost is in order of $O(L^2(I+1)^2)$. Secondly, the system identified using the algorithm has erratic local behavior and the local models have low local interpretability [4], [7]. This jeopardizes the initial motivation of using the TSK model to gain insights into the local models, such that interpretability of the TSK model is possible [4]. Therefore, the parameter learning algorithm of TSK model is a multi-objective identification problem; which would require a balance between good global learning accuracy and local model interpretability [7].

Fig. 4. Time-complexity of FITSK with recursive least-squares learning.

A simple experiment was performed to demonstrate the exponential growth in computational complexity with FITSK when adopting the recursive least-squares parameter learning (described in (2.23)). Consider a multidimensional function of the form described by (2.24)

$$y = \sum_{i=1}^{Dim} x_i^2 \quad (2.24)$$

where Dim is the dimension of the function.

Fig. 4 illustrates the time-complexity of first-order FITSK with recursive least-squares algorithm for parameter learning of function (2.24) with different number of input dimension. There are three labels for each dimension at layer 2 of FITSK in this experiment. The time-complexity of the learning process suffers from an exponential order as the dimension increases. This demonstrates the seriousness of the *curse-of-dimensionality* problem [10] in the FITSK model with recursive least-squares learning. Similar significance of the problem can be easily extensible to the general TSK model using a global learning algorithm, for example, the ANFIS model [9], [21].

With the hindsight from the above discussion, the modeling of the first-order FITSK should be positioned as a multi-objective problem; namely: 1) relaxing the computational complexity; 2) improving the local model interpretability; and 3) maintaining the global learning accuracy. These motivate the derivation of a localized version of recursive least-squares, which is design to achieve the three criteria, and simultaneously being an online algorithm with fast convergence to optimal performance.

Consider a local approximation of an arbitrary local model (denoted by (l, m)) in layer 5 of FITSK. Each training example is being influenced by the weight w_l . Thus, the local approximation can be represented as input-output relationships as shown in (2.25)

$$\begin{bmatrix} w_l^{(1)} & w_l^{(1)}x_1^{(1)} & w_l^{(1)}x_2^{(1)} \\ w_l^{(2)} & w_l^{(2)}x_1^{(2)} & w_l^{(2)}x_2^{(2)} \\ \vdots & \vdots & \vdots \\ w_l^{(p)} & w_l^{(p)}x_1^{(p)} & w_l^{(p)}x_2^{(p)} \\ \vdots & \vdots & \vdots \\ w_l^{(P)} & w_l^{(P)}x_1^{(P)} & w_l^{(P)}x_2^{(P)} \end{bmatrix} \begin{bmatrix} b_{0,(l,m)} \\ b_{1,(l,m)} \\ b_{2,(l,m)} \end{bmatrix} = \begin{bmatrix} w_l^{(1)}d^{(1)} \\ w_l^{(2)}d^{(2)} \\ \vdots \\ w_l^{(p)}d^{(p)} \\ \vdots \\ w_l^{(P)}d^{(P)} \end{bmatrix} \quad (2.25)$$

in a form of that can be identified by the least-squares algorithm, $A_{(l,m)}B_{(l,m)} = C_{(l,m)}$.

Letting $a_{p,(l,m)}$ be the p^{th} row vector of matrix $A_{(l,m)}$, and $a_{p,(l,m)} = [1 \ x_1^{(p)} \ x_2^{(p)}]$. To simplify the discussion, the subscript

(l, m) is omitted from (2.26) onwards since each local model is updated locally

$$\begin{aligned} a'_p &= \begin{bmatrix} w^{(p)} & w^{(p)}x_1^{(p)} & w^{(p)}x_2^{(p)} \end{bmatrix} \\ &= w^{(p)} \begin{bmatrix} 1 & x_1^{(p)} & x_2^{(p)} \end{bmatrix} = w^{(p)} a_p \end{aligned} \quad (2.26)$$

then B can be iteratively estimated by substituting the solution from (2.23) with (2.26)

$$\begin{aligned} B_{p+1} &= B_p + \left(w^{(p)} \right)^2 S_{p+1} a_{p+1}^T \left(d^{(p+1)} - a_{p+1} B_p \right) \\ S_{p+1} &= S_p - \frac{\left(w^{(p)} \right)^2 S_p a_{p+1}^T a_{p+1} S_p}{1 + \left(w^{(p)} \right)^2 a_{p+1} S_p a_{p+1}^T} \\ p &= 0, 1, \dots, P-1 \end{aligned} \quad (2.27)$$

and the local learning error is shown in (2.28)

$$\delta_{p+1} = \left(d^{(p+1)} - a_{p+1} B_p \right) \quad (2.28)$$

which is an approximation of the global learning error as

$$\delta_{p+1} = \left(d^{(p+1)} - y^{(p+1)} \right) \quad (2.29)$$

Therefore, (2.27) can be simplified into

$$\begin{aligned} B_{p+1} &= B_p + \left(w^{(p)} \right)^2 S_{p+1} a_{p+1}^T \delta_{p+1} \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{\frac{1}{\left(w^{(p)} \right)^2} + a_{p+1} S_p a_{p+1}^T} \\ p &= 0, 1, \dots, P-1 \end{aligned} \quad (2.30)$$

which is the optimal recursive solution to the least-squares problem $AB = C$. Therefore, the mean-squared-error (MSE) criterion is the optimal objective function for this least-squares-minimization problem. Subsequently, the optimal locally weighted MSE criterion of a local model is computed with (2.31)

$$Local\_MSE = \frac{1}{P} \sum_{p=1}^P \left(w^{(p)} \delta_p \right)^2. \quad (2.31)$$

The local_MSE criterion is the objective measure of the learning accuracy of a locally weighted local model. The total locally weighted MSE of all the local models is the sum of all local_MSEs.

$$Local\_MSE_{total} = \frac{1}{P} \sum_{l=1}^L \sum_{m=1}^M \sum_{p=1}^P \left(w_l^{(p)} \delta_{(l,m),p} \right)^2 \quad (2.32)$$

where

$L \times M$ total number of local models;
 (l, m) indexing of the local model.

On the other hand, (2.30) can be generalized with a degree of localization, ld

$$\begin{aligned} B_{p+1} &= B_p + \left(w^{(p)} \right)^{ld} S_{p+1} a_{p+1}^T \delta_{p+1}, \\ S_{p+1} &= S_p - \frac{S_p a_{p+1}^T a_{p+1} S_p}{\frac{1}{\left(w^{(p)} \right)^{ld}} + a_{p+1} S_p a_{p+1}^T} \\ p &= 0, 1, \dots, P-1 \end{aligned} \quad (2.33)$$

with initial conditions of $B_0 = \mathbf{0}$ and $S_0 = \gamma \mathbf{I}$, and a degree of localization of value 2 will always result in the optimal error as shown in (2.31).

Equation (2.33) is a proposed localized version of recursive least-squares algorithm. Each local model has its local copy of S matrix and B vector. The parameter tuning is computed locally instead of globally with respect to all the local models. The influence of an individual local model for a training example is measured with its weight. The degree of localization indicates the locality of the learning to the local model, and is indirectly proportional to the influence of training examples that are further away in the input spaces to a particular local model. A higher degree of localization will often result in better interpretability of local models.

Consider that K number of data points nearest to the centre of a local model in the local subspaces is responsible for influencing the interpretability of the local model. Then, the interpretability of the local models is measured with a K -central-nearest MSE (KCN_MSE) criterion, which is computed by (2.34)

$$KCN\_MSE = \frac{1}{N \times K} \sum_{n=1}^N \sum_{k=1}^K \delta_{n,k}^2 \quad (2.34)$$

where

$\delta_{n,k}$ error of the n^{th} local model w.r.t. the k^{th} data point;
 K total number of data points under consideration;
 k index.

The level of interpretability of local models is related to the degree of localization, and will be demonstrated with a simulation in Section III-C using the criterion described in (2.34).

The first-order FITSK fuzzy model with local recursive least-squares of I -input, L -rules will have L number of S matrices, each with dimension of $(I+1)^2$. The local recursive least-squares is repeated for each local model during the training process, the computational cost is in order of $O(L(I+1)^2)$. Furthermore, the weight of most local models is zero or near-to-zero for high-dimensional FITSK model during a particular training instance. The computational cost can be further reduced by bypassing the training of these near-to-zero weight local models. This significantly reduces the computational and space complexities, especially for higher-dimension FITSK model.

Fig. 5 illustrates the time-complexity of a first-order FITSK with the local recursive least-squares algorithm for the parameter learning of the function described by (2.24) with different number of input dimension. There are 3 labels for each dimension at layer 2 of FITSK in this experiment. The time-complexity of the learning process is still of exponential order. However, the computational cost has been significantly reduced by comparing with the time-complexity of FITSK with global recursive least-squares as shown in Fig. 4. By comparing Fig. 5 and Fig. 4, the computational cost for a 10-dimensional data with local recursive least-squares algorithm is lower than the cost of the global recursive least-squares algorithm with 5-dimensional data. This demonstrates that the FITSK model

Fig. 5. Time-complexity of FITSK with local recursive least-squares learning.

Fig. 6. Flowchart on 3-criteria model selection of FITSK.

adopting the local recursive least-squares algorithm has better immunity to the curse-of-dimensionality.

This research focuses on providing a general FITSK framework such that a specific model can be formulated with ease depending on different design criteria and considerations. The criteria for considerations are the interpretability, accuracy and dimensionality of the system; all these criteria are design or problem-dependant. Formulating this design problem as a multi-objective optimization problem is not justifiable practically since these objectives are system-dependant. Therefore, a guideline is provided to cater for the selection of FITSK model based on these three criteria.

Fig. 6 illustrates the design guideline on the model selection of FITSK based on three criteria. If the interpretability of a learning system is of the highest priority, a first-order FITSK model with localized version of recursive least-squares is adopted with a high degree of localization. This is to ensure that the learning is performed locally for each regional subspace. However, a zero-order FITSK model is chosen when both interpretability and accuracy is not important, and a straightforward and fast adaptive system is required. Finally, a first-order FITSK model is necessary on the condition that accuracy remains an important criterion. The global recursive least-squares learning algorithm is adopted when the system has low dimensionality since this approach is computational costly. On the other hand,

for higher dimensional system, the local recursive least-squares is chosen, and since interpretability is not a main consideration, a low degree of localization will adequately accommodate such need.

III. SIMULATION RESULTS AND ANALYSIS

Five nonlinear estimation tasks are studied in this section; the first three benchmarking tasks are taken from [22], they are; namely: 1) a nonlinear system; 2) a human-operated chemical plant; and 3) a daily price of a stock. These are benchmarked against five models [22] in a thesis [23] and the results are summarized in Section III-A. The fourth task is a Mackey–Glass time series prediction taken from [9], and is compared against the ANFIS model and TSK⁰-FCMAC model as proposed by [24]. The last task is a motorcycle crashing simulation taken from [18], to comprehend the interpretability issue [5] of the first-order FITSK model. The detail descriptions of each task are to be referred to each respective research [9], [18], [22].

The specifications of these tasks are shown in Table I.

The third column of Table I shows the selected dimensions of the original problem via structure identification [22]. The MSE and Pearson correlation coefficient are used to benchmark the results, The MSE measures the performance accuracy in terms of training or testing result. The Pearson product moment correlation coefficient is a dimensionless index that ranges from -1.0 to 1.0 and reflects the extent of a linear relationship between two sequences of data.

Defining MSE and root-MSE between two data vectors as (3.1)

$$MSE(\vec{x}, \vec{y}) = \frac{1}{n}(\vec{x} - \vec{y})^T(\vec{x} - \vec{y})$$

$$RMSE(\vec{x}, \vec{y}) = \sqrt{MSE(\vec{x}, \vec{y})} \quad (3.1)$$

where

MSE MSE function;
 $RMSE$ root-MSE function;
 $\vec{x}, \vec{y}$ two data vectors;
 n number of elements in the vector.

The Pearson correlation coefficient is defined in (3.2)

$$R(\vec{x}, \vec{y}) = \frac{C(\vec{x}, \vec{y})}{\sqrt{C(\vec{x}, \vec{x})C(\vec{y}, \vec{y})}} \quad (3.2)$$

where

R Pearson correlation coefficient function;
 $\vec{x}, \vec{y}$ two sequence of data vectors;
 $C(.)$ covariance between two data vectors.

The two data vectors for performance evaluations are the desired and the actual outputs from the model respectively. The FITSK⁰ model refers to the zero-order FITSK model with a

TABLE II
RELATIVE MSE FOR THREE TASKS

MSE	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
Task 1	0.40	0.90	0.54	1.00	0.82	0.39
Task 2	0.15	0.33	1.00	0.34	0.13	0.02
Task 3	0.56	0.21	1.00	0.24	0.55	0.20

TABLE III
RELATIVE PEARSON CORRELATION COEFFICIENT FOR THREE TASKS

R	Sugeno's P&P-G	Sugeno's P	Sugeno's P-G	Mamdani's	Turksen's IVCRI	FITSK ⁰
Task 1	0.98	0.66	1.00	0.58	0.72	0.98
Task 2	0.98	0.94	0.99	0.94	1.00	1.00
Task 3	0.80	1.00	0.79	0.98	0.75	1.00

learning rate of 0.2, and standard deviation of 0.4247 times the resolution. The FITSK¹-GRLS denotes the first-order FITSK model with global recursive least-squares learning (based on (2.23)), and FITSK¹-LRLS represents the first-order FITSK model with local recursive least-squares learning (based on (2.33)).

A. Nakanishi's Nonlinear Estimation Tasks

The three nonlinear systems were described in [22]. They are; namely: 1) a nonlinear system; 2) a human operated chemical plant; and 3) a daily price of a stock. The result of FITSK⁰ model is compared against five other models, the Sugeno's P&P-G, P, P-G, Mamdani's model, and Turksen's IVCRI model. These models differ in terms of modeling and theoretical basis, the details can be found in [22]. Turksen's PVAAR and IVAAR models are not included in the evaluation due to the large number of missing values in their results. This is because the performance results of these two models will significantly degrade by considering such missing values to be out-of-range, and are substituted with default value of zeros.

Detailed result discussions of these three tasks are presented in [23]. Table II summarizes the performances of models in the three tasks with MSE. The table normalizes each MSE with the best result within the row. Summarizing all the three tasks as discussed above, FITSK⁰ is able to achieve the lowest MSE for all the tasks with significant accuracy improvements over all the other models. Table III summarizes the performances of models in the three tasks with Pearson correlation coefficient. The table normalizes each coefficient with the best result within the row. As shown in the table, the FITSK⁰ model is the only model that is capable of achieving the optimal performances for all the three tasks. Models such as Sugeno's (P), (P-G) or Turksen's (IVCRI) performed well for a particular task and but not the others. This demonstrates that the FITSK⁰ model is outperforming other popular models despite being a simple zero-order local model TSK network for nonlinear system estimation.

B. Mackey–Glass Time Series Prediction

The dynamics of Mackey–Glass differential delay equation is defined in (3.3).

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \quad (3.3)$$

This time series is a popular benchmark problem considered by several researchers [9], [24].

The time series is obtained similarly as in Jang's thesis [9]. The fourth-order Runge-Kutta method was applied to compute the numerical approximation with time step of 0.1, initial condition $x(0) = 1.2$, $\tau = 17$, and $x(t) = 0$ for $t < 0$, for a time period of $0 \leq t \leq 2000$. From the computed series, 1000 input-output data pairs from $118 \leq t \leq 1117$ were extracted with the following format: $[x(t-18), x(t-12), x(t-6), x(t); x(t+6)]$. The goal of the task is to use known values of the time series of past 18, 12, 6, and current time to predict the 6th instance ahead in time. The first 500 data pairs are used as training datasets while the remaining 500 data pairs are used as testing datasets.

The models to be studied are TSK⁰-FCMAC [24], ANFIS [9], FITSK⁰, FITSK¹-GRLS and FITSK¹-LRLS. FITSK¹-LRLS implements the first-order FITSK model with the proposed local recursive least-squares approach with a localization factor of 9. The models to be compared consist of TSK-type fuzzy models of zero-order (TSK⁰-FCMAC, FITSK⁰), and first-order (ANFIS, FITSK¹-GRLS, FITSK¹-LRLS).

The training datasets were randomly permuted into 100 different training datasets in TSK⁰-FCMAC simulation, while the other models had three different permutations of training datasets. All the models are running on the same computer platform, with debug version of executables being generated from C++ source codes (C for ANFIS).

Table IV lists the testing results of Mackey–Glass series for the various models. The training process lasted for 20 epochs for the zero-order models. The second column of the table lists the number of labels of the respective models for each input dimension at the layer 2 of the models; and the number of rules of the models are computed as (no. of fuzzy labels)⁴ since there are four input dimensions. TSK⁰-FCMAC had 218 rules as those zero-weight rules were not included in the computation of the total number of rules.

Comparing the zero-order models, the FITSK⁰ is able to achieve a lower testing error despite using lesser number of rules than the TSK⁰-FCMAC, which is due to the Gaussian membership function adopted in layer 2 of the model. The average training time of FITSK⁰ is acceptably low despite the usage of

TABLE IV
TESTING RESULTS OF MACKEY-GLASS SERIES

Model	No. of labels (layer 2)	Average RMSE	Std. dev. (RMSE)	Training epochs	Avg training time (s)	No. of rules
TSK ⁰ -FCMAC	4	0.02229	0.00468	20	1.92	218
FITSK ⁰	3	0.01842	0.00004	20	0.99	81
ANFIS	2	0.00295	0.00000	1	0.27	16
FITSK ¹ -GRLS	2	0.00321	0.00000	1	0.33	16
FITSK ¹ -GRLS	3	0.00139	0.00000	1	9.78	81
FITSK ¹ -LRLS	2	0.03070	0.00000	1	0.09	16
FITSK ¹ -LRLS	3	0.01926	0.00000	1	0.13	81

Gaussian membership function, which is computational costly comparing with the trapezoidal or triangular fuzzy functions in TSK⁰-FCMAC.

Next, it is appropriate to compare the two first-order models with two linguistic labels, the ANFIS model at the fourth row and FITSK¹-GRLS model at the fifth row. It is shown that both models have similar accuracy and training time since they both adopt the global recursive least-squares learning algorithm. The minor differences may be due to programming differences. The 6th row demonstrates the FITSK¹-GRLS model with three linguistic labels at layer 2, and thus larger number of rules (local models). As expected, the accuracy improves significantly over the previous 2 models as the number of local models increases. However, the drawback of this is the significant growth in computational cost, which increases from 0.33 s to 9.78 s.

The next two models are the FITSK¹-LRLS with two and three linguistic labels in each dimension; both adopting local recursive least-squares parameter learning with degree of localization being set to numeric value 2. The computational time of these two models demonstrate a significant reduction comparing with models with global recursive least-squares approach. On the other hand, the growth in computational cost due to the increased number of local models is insignificant (from 0.09 s to 0.13 s). This clarifies that the parameter learning algorithm with local recursive least-squares approach is insignificantly affected by the curse-of-dimensionality having in the global recursive least-squares approach. Moreover, by using the recursive least-squares parameter learning approaches, all the first-order models are able to achieve fast learning convergence within a single training epoch. This inherits the online learning capability from the recursive least-squares parameter identification technique. The online learning capability is made possible by discarding each training instances after it is processed without repeatedly reassessing it, and the weights are subsequently updated based on the current training instances [25].

C. Motorcycle Crashing Simulation

The final task is the motorcycle crashing simulation data taken from [18], which has been briefly introduced in Section II-B. The data is revisited here to illustrate the interpretability issue in the first-order TSK model. The same dataset has been used in [5] to study the interpretability issue of TSK fuzzy model.

Firstly, eight linguistic labels in layer 2 of the FITSK model are equally position across the space of range [2.4 57.6]. The standard deviation for the Gaussian membership function is

chosen to be the minimum, which is 0.4247 of the resolution, as shown in (2.16). The global MSE of the system identified is 459.126 by using the global recursive least-squares parameter identification [(2.23)]. The global learning error is slightly better than the one identified by Yen's model [5], which is 460.62. Yen adopted a singular value decomposition method to identify redundant fuzzy partitions using a firing strength matrix. However, the method is a form of unsupervised clustering; no output value or output performance indication has been used to influence the result of clustering. Thus, the method could only identify the redundant fuzzy partitions among all the fuzzy partitions, but could not recognize the relevancy of the fuzzy partitions with respect to the output of the system. It can be examined from the result in [5] that, the fuzzy partitions identified by Yen's model is not optimal; i.e., there should be a fuzzy partition at a time near-to 26 ms since the output around this region changes drastically. However, the method is unable to monitor such drastic transition at the output space. Thus, Yen's fuzzy partition method performed no better than an equally spaced partition with a fuzzy input formulation as in the FITSK model. This is supported by the higher learning accuracy of the FITSK model than the one by Yen's model.

The interpretability of the first-order FITSK local models is investigated in Fig. 7. Fig. 7 illustrates the learning result of FITSK¹ with different settings. The original dataset is represented using black dots in the figure; the learnt-models are plotted with solid curves; and the corresponding eight local models are indicated by straight bold lines in the figure and denoted numerically from (1) to (8). Fig. 7(a) represents the training result of FITSK¹-GRLS, which is the first-order FITSK model with global recursive least-squares as the parameter learning algorithm. Fig. 7(b), (c), and (d) are first-order FITSK models with local recursive least-squares (FITSK¹-LRLS), the degree of localization of these models are 2, 6, and 10, respectively. By observation, it is easy to infer that the learnt systems have significant differences in the first, second, and fourth local models given different system settings. The global learning in Fig. 7(a) has good global learning error, but the interpretation of local models (1) and (2) are inappropriate. On the other extreme, the local models (1) and (2) in Fig. 7(d) have best interpretability given a high degree of localization ($ld = 10$). Local model (4) in Fig. 7(d) is misinterpreted to the neighboring region; however, if the observation is focused on a small regional subspace, it can be argued that the output data values are almost constant within this region. This coincides with the learning result of local model (4) in

Fig. 7. Global and local learning results of first-order FITSK models. (a) Global learning (FITSK¹-GRLS); (b) local learning (FITSK¹-LRLS, Id = 2); (c) local learning (FITSK¹-LRLS, Id = 6); (d) local learning (FITSK¹-LRLS, Id = 10).

Fig. 7(d), which is almost a constant term model. The results of Fig. 7(b) and (c) are in between the two extreme cases in Fig. 7(a) and (d), this is made possible with smaller settings of degree of localization for these models. By visual inspection, if the influence is confined to a regional subspaces proportional to the weight (matching degree) of the local models as shown in Fig. 7(b), then the FITSK¹-LRLS model with degree of localization 2 has the best local learning accuracy. The local models for this model have a good balanced between learning accuracy and model interpretability. A FITSK¹-LRLS with degree of localization equals 2 has optimal locally weighted MSE as formulated in (2.32).

The learning results of different FITSK¹ settings are illustrated with four different error criteria in Fig. 8. The criteria are the *Global_MSE* [(3.1)], *Local_MSE2* [(2.32)], *2CN_MSE* [(2.34) with $K = 2$], and *Local_MSE1* being defined by [8], as shown in (3.4). The *Local_MSE1* is similar to *Local_MSE2* except that the weight of *Local_MSE2* is order of 2

$$Local\_MSE1 = \frac{1}{P} \sum_{n=1}^N \sum_{p=1}^P w_n^{(p)} \delta_{n,p}^2. \quad (3.4)$$

The learning errors of FITSK¹-GRLS are plotted in $ld = 0$, and $ld = [1, 10]$ represents the FITSK¹-LRLS models with

Fig. 8. Learning errors of FITSK¹ with different degree of localization.

degree of localization ranging from 1 to 10 accordingly. As shown with the *Global_MSE* criterion, the FITSK¹ with global recursive least-squares learning has the best global learning accuracy. However, the accuracy gradually reduces

as the degree of localization increases. On the other hand, the *Local_MSE1* and *Local_MSE2* criteria demonstrate that the local recursive least-squares learning with degree of localization 2 has the optimal locally weighted LSE. This is supported by the observation from Fig. 7(b), which demonstrates that a degree of localization of 2 will result in FITSK¹ model with a balanced learning accuracy and local model interpretability. The *2CN_MSE* criterion is the strictest formulation of the general *KCN_MSE* criterion. It uses the two data points nearest to the centre of a local model to measure the interpretability of a local model. The *2CN_MSE* criterion illustrates that, the interpretability of the local models increases as proportional to the degree of localization. This coincides with Fig. 7(d), which shows that the local models have best interpretability to the smallest confined region at the centre of the local models.

These conclude that (2.34) is a good indicator of interpretability, and (2.32) is a good criterion to measure the locally weighted least-squares accuracy of the local models. The simulation demonstrates that the FITSK¹ model with local recursive least-squares can be easily reformulated to have different degree of interpretability and local model accuracy by varying the degree of localization; and is flexible to the system designer based on different design considerations.

IV. CONCLUSIONS

This paper proposes a generic TSK framework, the FITSK framework. The framework is subsequently mapped to a zero-order FITSK model and a first-order FITSK model. The zero-order and first-order FITSK models assume online parameter learning formulae. In this way, the FITSK models are suitable for adaptive system learning and nonlinear system estimation.

The key strength of the FITSK framework is the flexibility of adopting different formulations and settings to achieve different design criteria. The three design criteria are 1) interpretability; 2) global accuracy; and 3) immunity to curse-of-dimensionality. The zero-order FITSK model is suitable for fast adaptive system without a major consideration of interpretability. The first-order TSK model with global recursive least-squares is appropriate for a low-dimensionality, low-interpretability system with optimal global learning accuracy. A proposed localized version of the recursive least-squares algorithm has been shown to achieve optimal accuracy for locally weighted LSE criterion. Furthermore, the local recursive least-squares has been demonstrated to have better immunity than the global recursive least-squares to the curse-of-dimensionality issue. The first-order FITSK model adopting this local recursive least-squares can be readily applied to higher-dimensionality system with lower computational resources comparing with the one using global recursive least-squares learning. The degree of localization in the local recursive least-squares provides a flexibility to balance the interpretability and system accuracy in the local models. A guideline is discussed based on these three design criteria for model selection of these FITSK models.

The performance of the FITSK models are evaluated using five nonlinear system simulations: 1) a nonlinear system; 2) human operation of a chemical plant; 3) daily price of a stock; 4) Mackey–Glass time series prediction; and 5) motorcycle

crashing simulation. The first three simulations demonstrate that the zero-order FITSK model has encouraging performances in terms of learning error and correlation measure. The fourth simulation highlights the computational advantages of the local recursive least-squares in the first-order FITSK model. The final simulation illustrates the modeling flexibility of the first-order FITSK model to tackle the balancing between local model interpretability and learning accuracy.

A possible future work is to automate the model selection of FITSK, which is described in Fig. 6. This is especially important in automatically determining whether the system is likely to suffer from the curse-of-dimensionality issue. Currently, this process is performed manually based on the subjective perception of the designer. A formal approach should be adopted with supporting studies in the future.

The online leaning capability of the FITSK is currently demonstrated with a benchmarking Mackey–Glass time series prediction problem. Further research is desired by applying the FITSK to several process control applications. These include the fluidised bed combustor and the three tanks flow system; such that the FITSK is able to be contrasted against other popular models such as the distributed logic processors [26]–[28] and the blackboard-based BBIPS model [29].

REFERENCES

- [1] M. Sugeno and K. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, p. 28, 1988.
- [2] T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," in *Proc. IFAC Symp. Fuzzy Information, Knowledge Representation and Decision Analysis*, 1983, pp. 55–60.
- [3] —, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Feb. 1985.
- [4] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control, and Information*. Englewood Cliffs, NJ: Prentice-Hall, 1999.
- [5] J. Yen, L. Wang, and C. W. Gillespie, "Improving the interpretability of TSK fuzzy models by combining global learning and local learning," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 4, pp. 530–537, Nov. 1998.
- [6] J. Abonyi and R. Babuska, "Local and global identification and interpretation of parameters in Takagi–Sugeno fuzzy models," in *Proc. FUZZ-IEEE 2000*, vol. 2, 2000, pp. 835–840.
- [7] T. A. Johansen, R. Shorten, and R. Murray-Smith, "On the interpretation and identification of dynamic Takagi–Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 3, pp. 297–313, Jun. 2000.
- [8] T. A. Johansen and R. Babuska, "Multiobjective identification of Takagi–Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 6, pp. 847–860, Dec. 2003.
- [9] J. S. Jang, "Neuro-Fuzzy Modeling: Architecture, Analyzes and Applications," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, 1992.
- [10] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
- [11] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Mach. Studies*, vol. 7, no. 1, 1975.
- [12] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
- [13] T. K. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, pp. 59–69, 1982.
- [14] W. L. Tung and C. Quek, "GenSoFNN: a generic self-organizing fuzzy neural network," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1075–1086, Sep. 2002.
- [15] M. Männle, "Identifying rule-based TSK fuzzy models," in *Proc. EUFIT*, Aachen, Germany, Sep. 1999.
- [16] C. C. Lee, "Fuzzy Logic in control systems: fuzzy logic controller: part I," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 404–418, Mar./Apr. 1990.

- [17] —, "Fuzzy logic in control systems: fuzzy logic controller: part II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 419–435, Mar./Apr. 1990.
- [18] W. Hardle, *Applied Nonparametric Regression*. Cambridge, MA: Cambridge Univ. Press, 1990.
- [19] C. T. Lin and C. S. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism on Intelligent System*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
- [20] X. Xu, H. He, and D. Hu, "Efficient reinforcement learning using recursive least-squares methods," *J. Artif. Intell. Res.*, vol. 16, pp. 259–292, 2002.
- [21] J. S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
- [22] H. Nakanishi, I. B. Turksen, and M. Sugeno, "A review and comparison of six reasoning methods," *Fuzzy Sets Syst.*, vol. 57, pp. 257–294, 1993.
- [23] K. H. Quah, "Fuzzy Modeling in Reinforcement Learning," Ph.D. dissertation, Sch. Comput. Eng., Nanyang Technological Univ., Singapore, 2005.
- [24] C. W. Ting, "Fuzzy Associative Memory Architecture," Ph.D. dissertation, Sch. Comput. Eng., Nanyang Tech. Univ., Singapore, 2002.
- [25] W. S. Sarle. (1997) Neural Network FAQ, Part 2 of 7: Learning. [Online] Available: <ftp://ftp.sas.com/pub/neural/FAQ2.html>
- [26] E. Ikonen and K. Najim, "Use of learning automata in distributed fuzzy logic processor training," *Proc. Inst. Elect. Eng., Control Theory Appl.*, vol. 144, no. 3, pp. 255–262, 1997.
- [27] K. Najim and E. Ikonen, "Distributed logic processors trained under constraints using stochastic approximation techniques," *IEEE Trans. Syst., Man, Cybern. A*, vol. 29, no. 4, pp. 421–426, Jul. 1999.
- [28] E. Ikonen, K. Najim, and U. Kortela, "Neuro-fuzzy modeling of power plant flue-gas emissions," *Eng. Applicat. Artif. Intell.*, vol. 13, pp. 705–717, 2000.
- [29] K. W. Oh and C. Quek, "BBIPS: a blackboard-based integrated process supervision," *Eng. Applicat. Artif. Intell.*, vol. 14, pp. 703–714, 2001.

nition.

Kian Hong Quah received the B.A.Sc degree in computer engineering in 2000 from Nanyang Technological University, Singapore, where he is currently pursuing the Ph.D. degree in the Centre for Computational Intelligence, School of Computer Engineering.

He was a Research Engineer with Hewlett-Packard (Singapore) Pte. Ltd. from 2000 to 2001. His current research interests are in the areas of neural-fuzzy systems, fuzzy inference system, reinforcement learning, feature subset selection, and pattern recog-

Chai Quek (M'97) received the B.Sc. degree in electrical and electronics engineering in 1986 and the Ph.D. degree in intelligent control in 1990 from Heriot Watt University, Edinburgh, U.K.

He is an Associate Professor and a member of the Centre for Computational Intelligence (formerly the Intelligent Systems Laboratory), School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include intelligent control, intelligent architectures, AI in education, neural networks, fuzzy neural systems, neurocognitive informatics, and genetic algorithms.

He is a member of the IEEE Technical Committee on Computational Finance.

ELSEVIER

Expert Systems with Applications xx (2005) 1–9

Expert Systems
with Applications

www.elsevier.com/locate/eswa

Maximum reward reinforcement learning: A non-cumulative reward criterion

K.H. Quah, Chai Quek *

Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University,
Blk N4 #2A-32, Nanyang Avenue, Singapore 639798

Abstract

Existing reinforcement learning paradigms proposed in the literature are guided by two performance criteria; namely: the expected cumulative-reward, and the average reward criteria. Both of these criteria assume an inherently present cumulative or additivity of the rewards. However, such inherent cumulative of the rewards is not a definite necessity in some contexts. Two possible scenarios are presented in this paper, and are summarized as follows. The first concerns with learning of an optimal policy that is further away in the existence of a sub-optimal policy that is nearer. The cumulative rewards paradigms suffer from slower convergence due to the influence of accumulating the lower rewards, and take time to fade away the effect of the sub-optimal policy. The second scenario concerns with approximating the supremum values of the payoffs of an optimal stopping problem. The payoffs are non-cumulative in nature, and thus the cumulative rewards paradigm is not applicable to resolve this. Hence, a non-cumulative reward reinforcement-learning paradigm is needed in these application contexts. A maximum reward criterion is proposed in this paper, and the resulting reinforcement-learning model with this learning criterion is termed the *maximum reward reinforcement learning*. The maximum reward reinforcement learning considers the learning of non-cumulative rewards problem, where the agent exhibits a maximum reward-oriented behavior towards the largest rewards in the state-space. Intermediate lower rewards that lead to sub-optimal policies are ignored in this learning paradigm. The maximum reward reinforcement learning is subsequently modeled with the FITSK-RL model. Finally, the model is applied to an optimal stopping problem with a nature of non-cumulative rewards, and its performance is encouraging when benchmarked against other model.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: Maximum reward; Reinforcement learning; Non-cumulative reward; FITSK-RL; Optimal stopping problem; Financial derivative pricing; Two-cycle task; Maximum reward-oriented behaviour

1. Introduction

Reinforcement learning is a trial and error learning method, where the agent tries to explore and maximize the numerical state values from the feedback it receives over the long run. The process that the agent follows is modeled as the *Markov decision process* (MDP). In the MDP context, the two most well-studied optimality criteria are the expected cumulative reward and the average reward criteria (Puterman, 1994). The expected cumulative reward criterion has been adopted as the de facto objective measure in reinforcement learning research (Bertsekas, & Tsitsiklis, 1996; Kaelbling, Littman & Moore, 1996; Rummery & Niranjan, 1994; Sutton, 1984, 1988; Sutton & Barto, 1998; Watkins, 1989). This criterion employs a

technique to exponentially discount the future rewards, such that the near-future rewards are more valuable than far-future rewards. However, some research has demonstrated that the expected cumulative reward criterion is unsuitable for some problems, particularly in the undiscounted reward domain (Schwartz, 1993). Some reinforcement learning methods based on the average reward criterion have been proposed to resolve the undiscounted reward reinforcement learning problem (Mahadevan, 1996; Schwartz, 1993; Tadepalli & Ok 1998). In the essence of this, both classes of methods employ some form of cumulative rewards criteria; the former with a reward discounting approach such that their infinite cumulative rewards are finite, and the latter with a reward averaging approach such that both short and long-term rewards are indistinguishable.

However, a central issue is a problem of whether such a cumulative rewards formulation is a definite necessary requirement in all-learning contexts. Is there any problem that the cumulative rewards formulation is unnecessary, and unjustifiable? This paper discusses two possible scenarios

* Corresponding author. Tel.: +65 6790 4926; fax: +65 6792 6559.

E-mail address: ashquek@ntu.edu.sg (C. Quek).

where the cumulative reward formulation is to be avoided for superior learning result. This is demonstrated in Sections 3 and 4. In Section 3, a two-cycle task from (Mahadevan, 1996) is discussed. One of the cycle leads to a sub-optimal policy and the other cycle leads to an optimal policy, but the rewards of the optimal policy is further away. Therefore, the focus of the learning is on how to bypass the sub-optimal policy in the search for an optimal policy. The classical Q-learning (Watkins, 1989) was demonstrated to suffer from this problem because it was being dragged down by the sub-optimal policy, causing it to be lagging behind from switching to a better policy. In Section 4, an optimal stopping problem arising from financial derivative pricing (Tsitsiklis & Van Roy, 1999) is discussed. This is a case where the cumulative rewards formulation is unjustifiable. On the other hand, this problem is concerned with finding the supremum of the payoff values of each state, which is a function of the expected discounted maximum rewards in a non-cumulative manner. Furthermore, a discounted reward is significant in this domain as the discounting operation represents a compensation of future rewards with respect to the drift rate of financial pricing. This represents a contrary argument against the claim by (Schwartz, 1993) that discounting of rewards is not necessarily important. Having these, a reinforcement learning formulation with non-cumulative *maximum reward* criterion is proposed to accommodate the learning needs of such tasks.

The maximum reward reinforcement-learning agent exhibits a maximum reward-oriented behavior. It attempts to search for the policy that leads to the largest reward; irregardless of intermediate lower rewards along the path. This positions it to have superior immunity to the existence of sub-optimal policies in the problem domain, such as the two cycles problem in Section 3. Furthermore, the maximum reward learning is demonstrated to be effective in approximating the future payoff in a financial pricing context. The learning system exhibits an asymptotic behavior of approximating the supremum value with the maximum reward formulation, as will be discussed in Section 4.

This paper is organized as follows. Section 2 describes the general formulation of the maximum reward reinforcement learning and its update formulae as part of the modular design of the earlier proposed generic reinforcement-learning framework from (Quah & Quek, submitted for publication). Section 3 studies the learning behavior of the maximum reward reinforcement learning using a two-cycle task, and compares against two existing reinforcement-learning models. Section 4 applies the proposed maximum reward reinforcement learning with its generic learning framework to an optimal stopping problem arising from the financial derivative pricing context. Section 5 concludes this paper.

2. Maximum reward reinforcement learning

This section formulates the maximum reward reinforcement learning. Section 2.1 provides some basic definitions from the existing research. Section 2.2 mathematically formulates the maximum reward reinforcement learning, discusses the three

existing reinforcement-learning models, and the equivalent form of these three models for the maximum reward reinforcement learning. Section 2.3 solves the modelling problem of the maximum reward reinforcement learning. This is made possible by mapping the maximum reward error update formula to a generic reinforcement-learning framework to inherit the modularity and functionality of the framework.

2.1. Background definitions

Consider a sequence of an infinite horizon *Markov decision process* (MDP) (Puterman, 1994) with discrete time step $t = 0, 1, 2, 3, \dots, \infty$. At each discrete time step, the agent receives some observable environment's states, $S_t \in S$, where S is the set of possible states. The agent implements a mapping from the state to probabilities of selecting each possible action, $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state s_t . The mapping is called the agent's policy, π_t , and is depicted by Eq. (2.1).

$$\text{Policy mapping; } \pi_t(s, a) = \Pr(a_t = a | s_t = s) \quad (2.1)$$

indicates the probability of choosing action a when the perceived state is s at time t . Due to the action a_t , the agent receives a numerical reward r_{t+1} , and observes a new state s_{t+1} at time $t+1$.

The goal of the learning algorithm is to maximize the successive total reward, or the *return* value as described in Eq. (2.2).

$$\text{Return; } R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2.2)$$

The discount rate, γ , determines the present value of future rewards, a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately. If $\gamma = 0$, then the agent's actions only influence the immediate reward, r_{t+1} , as in most supervised learning agent. As γ approaches 1, the agent will take future rewards more seriously, or become more farsighted.

2.2. Formulation of maximum reward reinforcement learning

This section defines a possible variation of return value, and extends the definition to various Bellman's optimality formulae. This is followed by the formulation of the maximum reward form of temporal difference learning models in reinforcement learning.

The return value is defined as the cumulative future rewards in Eq. (2.2) where the agent is learning to maximize the return value. However, if the objective of the agent is to identify the largest reward within the value space, then the return value formulae in Eq. (2.2) can be suitably modified into Eq. (2.3).

$$R_t = \max(r_{t+1}, \gamma r_{t+2}, \gamma^2 r_{t+3}, \dots) = \max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+1}) \quad (2.3)$$

The maximum reward agent will attempt to discover the shortest path that lead to the highest reward, without

concerning too much on the intermediate lower rewards. This constructs the fundamental behavior exhibited in the maximum reward reinforcement learning.

Given any state, s , and action, a , the probability of each possible next state, s' , or the *transition probability* from state s to state s' is given by Eq. (2.4).

$$P_{ss'}^a = \Pr(s_{t+1} = s' | s_t = s, a_t = a) \quad (2.4)$$

Similarly, the expected value of the next reward is given by Eq. (2.5).

$$R_{ss'}^a = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s') \quad (2.5)$$

where $E(\cdot)$ is the expectation function.

Given a policy, π , which maps state s to an action a , the *value* of the state s under policy π , denoted by $V^\pi(s)$, is defined as the expected *return* at state s by following the same policy in all subsequent states. This is described in Eq. (2.6).

$$\begin{aligned} V^\pi(s) &= E_\pi(R_t | s_t = s) = E_\pi \left(\max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+1}) | s_t = s \right) \\ &= E_\pi \left(\max \left(r_{t+1}, \gamma \max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+2}) \right) | s_t = s \right) \\ &= \sum_{\forall a \in A(s)} \pi(s, a) \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma V^\pi(s'))] \end{aligned} \quad (2.6)$$

where $E_\pi(\cdot)$ is the expectation function with a given policy π .

Similarly, the *action-value function for policy π* is defined as the value of taking action a in state s under a policy π , denoted as $Q^\pi(s, a)$, and described by Eq. (2.7).

$$\begin{aligned} Q^\pi(s, a) &= E_\pi(R_t | s_t = s, a_t = a) \\ &= E_\pi \left(\max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+1}) | s_t = s, a_t = a \right) \\ &= E_\pi \left(\max \left(r_{t+1}, \gamma \max_{\forall k \in [0, \infty]} (\gamma^k r_{t+k+2}) \right) | s_t = s, a_t = a \right) \\ &= \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma V^\pi(s'))] \\ &= \sum_{\forall s' \in S'} P_{ss'}^a \left[\max \left(R_{ss'}^a, \gamma \sum_{\forall a' \in A'(s')} \pi(s', a') Q^\pi(s', a') \right) \right] \end{aligned} \quad (2.7)$$

Consider an *optimal policy*, π^* , where,

$$V^{\pi^*}(s) \geq V^\pi(s), \quad (2.8)$$

for all s and π .

An optimal policy produces the largest state value for all s among all the policies. Define the *optimal state-value function*, V^* , as Eq. (2.9).

$$\begin{aligned} V^*(s) &= \max_{\pi} V^\pi(s) = \max_a Q^{\pi^*}(s, a) \\ &= \max_a \sum_{\forall s' \in S'} P_{ss'}^a [\max(R_{ss'}^a, \gamma V^*(s'))] \end{aligned} \quad (2.9)$$

Similarly, the *optimal action-value function*, Q^* , is described by Eq. (2.10).

$$\begin{aligned} Q^*(s, a) &= \max_{\pi} Q^\pi(s, a) \\ &= E(\max(r_{t+1}, \gamma V^*(s_{t+1})) | s_t = s, a_t = a) \\ &= E \left(\max \left(r_{t+1}, \gamma \max_{a'} Q^*(s_{t+1}, a') \right) | s_t = s, a_t = a \right) \\ &= \sum_{\forall s' \in S'} P_{ss'}^a \left[\max(R_{ss'}^a, \gamma \max_{a'} Q^*(s', a')) \right] \end{aligned} \quad (2.10)$$

Eqs. (2.9) and (2.10) represent the maximum reward Bellman optimality equations for V^* and Q^* .

The temporal difference error equations for the three popular reinforcement learning models are listed in Eqs. (2.12), (2.13) and (2.14). These models are the temporal difference learning (Sutton, Sutton & Barton, 1988, 1998), the SARSA learning (Rummery & Niranjan, 1994), and the Q-learning (Watkins, 1989), respectively.

The general TD learning error follows from the error function in Eq. (2.11).

$$\delta_t = R_t - E(R_t) \quad (2.11)$$

Therefore, the TD error of temporal difference learning for a state at time t is given in Eq. (2.12).

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \quad (2.12)$$

For on-policy SARSA learning, the TD error is described in Eq. (2.13).

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \quad (2.13)$$

Similarly, for off-policy Q-learning, the TD error is described in Eq. (2.14).

$$\delta_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \quad (2.14)$$

Each popular reinforcement-learning model can be modified into a respective form of maximum reward reinforcement learning model. It can be formulated by modifying the return value to take on the maximum reward, as shown in Eq. (2.3). Following these, the TD error formulae for maximum reward reinforcement learning models are shown in Eq. (2.15). The subscript M denotes that the objectives of these learning models are the maximum reward values instead of the cumulative reward values.

$$\text{TD}_M \text{ learning : } \delta_t = \max(r_{t+1}, \gamma V_t(s_{t+1})) - V_t(s_t)$$

$$\text{SARSA}_M \text{ learning : } \delta_t$$

$$= \max(r_{t+1}, \gamma Q_t(s_{t+1}, a_{t+1})) - Q_t(s_t, a_t) \quad (2.15)$$

$$\text{Q}_M \text{ - learning : } \delta_t$$

$$= \max(r_{t+1}, \gamma \max_a Q_t(s_{t+1}, a)) - Q_t(s_t, a_t)$$

Further extensions of these maximum reward TD error formulae to the respective reinforcement learning models

Fig. 1. A generic reinforcement learning framework.

are straightforward and will be discussed in Section 2.3. The differences in the behaviour and the strengths of the maximum reward reinforcement learning over existing reinforcement learning models will be demonstrated with simulations and examples in the latter part of this paper.

2.3. Mapping to a generic reinforcement learning framework

This subsection extends the maximum reward reinforcement learning update formulae (Eq. (2.15)) into the generic reinforcement learning framework proposed in (Quah & Quek, submitted for publication). The block diagram of the generic agent of the framework is shown in Fig. 1.

The generic agent adopts a modular design with three modules; namely: the function approximator, the policy generator, and the temporal difference (TD) generator. The function approximator module predicts the value functions based on the state information. The policy generator module generates an appropriate action and the corresponding action-value. The TD generator module subsequently formulates the temporal difference error based on the reinforcement signal, the current action-value, and the successive action-value. Consequently, the temporal difference error is feedback to the function approximator module to adjust the tunable parameters. Two motivations for having this modular design are the functional generality or abstraction, and the functional independence. Having generic functionality for each respective module enables the framework to be abstractly defined, and thus easily extendable to different reinforcement learning model such as TD learning, SARSA learning, or Q-learning. Moreover, the functional independence ensures that the modules are not tightly dependant on each other, and thus reformulation or implementation of a single module would not affect the general operations of the other modules. The discussion in this paper is limited to the following extent; the TD generator module is being generalized to incorporate the learning of maximum reward update formulae. Please refer

to (Quah and Quek, submitted) on the definitions of the function approximator and the policy generator modules.

The TD generator computes the temporal difference error as a function of the immediate reward, current action-value, and successive action-value estimation, as shown in Eq. (2.16).

$$\text{TD error : } \delta_t = f_{\text{TD}}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) \quad (2.16)$$

where $f_{\text{TD}}(\cdot)$ is a generic function of the TD generator module.

The generic TD function is extended to the TD error of the cumulative reward formulation with Eq. (2.17).

$$\begin{aligned} \text{Cumulative reward TD error : } \delta_t &= f_{\text{TD}}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) \\ &= r_{t+1} + \gamma Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \end{aligned} \quad (2.17)$$

Similarly for the maximum reward formulation, the generic TD function adopts the formula as shown in Eq. (2.18).

$$\begin{aligned} \text{Non-cumulative(maximum)reward TD error : } \delta_t &= f_{\text{TD}}(r_{t+1}, Q_t(s_{t+1}, a'), Q_t(s_t, a_t)) \\ &= \max(r_{t+1}, \gamma Q_t(s_{t+1}, a')) - Q_t(s_t, a_t) \end{aligned} \quad (2.18)$$

The TD error formulae in Eqs. (2.17) or (2.18) can be easily mapped to TD learning, SARSA learning or Q-learning by adopting different action selection strategies in the policy generator module, as demonstrated in (Quah & Quek, submitted for publication). Furthermore, the maximum reward reinforcement learning can be directly incorporated into the FITSK-RL model (Quah & Quek, submitted) by adopting a zero-order FITSK model (Quah & Quek, submitted for publication) as the function approximator module. This enables the value function to be effectively approximated with a linear approximator (linear to the tunable parameter) in a systematic manner, and extendable to learning with the eligibility trace formulations (Sutton, 1984). Please refer to (Quah & Quek,

submitted for publication) on the technical details of formulating the FITSK-RL model.

3. Comparison with Q-learning and R-learning

This section compares the proposed non-cumulative, maximum reward reinforcement learning with existing learning models with inherent cumulative rewards reinforcement learning models, particularly the Q-learning (Watkins, 1989) and the R-learning (Schwartz, 1993). Q-learning is an off-policy, model-free reinforcement learning model, and is shown to be effective in control task learning with discounting of cumulative rewards. The learning equation for Q-learning is described by Eq. (2.14). The basic idea of formulating a Q-learning control is to adopt a greedy policy as the estimation policy, while simultaneously exploring the search space with a non-greedy behaviour policy. Similarly, R-learning is an off-policy, model-free model. However, the objective of R-learning is to maximize the performance based on the undiscounted cumulative rewards value, instead of the discounted cumulative rewards value described in Eq. (2.2). The undiscounted cumulative rewards value of a process starting at time t is shown in Eq. (3.1).

$$\rho_t = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^n r_{t+k+1} \quad (3.1)$$

where ρ_t is the average undiscounted cumulative rewards at time t . It was shown in (Schwartz, 1993) that if a policy gives rise to an *ergodic* Markov chain, the average reward values for all the states in that ergodic Markov chain are identical (Eq. (3.2)).

$$\rho = \rho_t, \quad \forall s_t \in S_E \quad (3.2)$$

where S_E is the set of Ergodic states.

The update formulae of R-learning are represented in Eq. (3.3). Please refer to (Schwartz, 1993) on the rational of these update formulae.

$$\begin{aligned} RV_{t+1}(s_t, a_t) &= RV_t(s_t, a_t) + \alpha_1(r_{t+1} - \rho_t) \\ &\quad + \max_a RV_t(s_{t+1}, a) - RV_t(s_t, a_t) \\ \rho_{t+1} &= \rho_t + \alpha_2 G(a_t)(r_{t+1} + \max_a RV_t(s_{t+1}, a) \\ &\quad - RV_t(s_t, a_t) - \rho_t) \end{aligned} \quad (3.3)$$

where,

$RV(\cdot)$ is a vector of realnumbers, corresponding to each state-action pair (s, a)
 α_1, α_2 are learning rate parameters, and
 $G(a_t)$ is a logic function with output 1 if a_t is greedily selected, and 0 otherwise.

Fig. 2 shows an example from (Mahadevan, 1996), which is an adaptation of the two cycles Markov decision process (MDP) task (Schwartz, 1993) with action choice at state 1 (Home) of the MDP. A robot following that MDP receives

Fig. 2. An MDP with two cycles.

reward of +5 at the Printer room (state 14) for a trip from Home to Printer room, and receives reward of +20 for servicing the Mailroom (state 10) that is further away. Learning simulations are performed for R-learning (Eq. (3.3)), Q-learning (Eq. (2.14)) and Q_M -learning (Eq. (2.15)), respectively. All the learning rates have numerical value 0.2, and all the behavior policies are ϵ -greedy (Sutton, 1998) with $\epsilon=0.05$.

Fig. 3 shows the performance comparison between R-learning, Q-learning and Q_M -learning. The results are averaged over 5 independent runs, each run consisted of 20,000 number of steps. The performance is measured with the average of rewards being accumulated over the past 1000 number of steps. The optimal path follows the cycle from 1-2-3-4-5-6-7-8-9-10, and repeats itself infinitely. Therefore, the maximum of the average reward value is 2, and is derived by averaging the largest reward, 20, over 10 actions for each cycle. The learning agent may often stay at the suboptimal policy, which follows the cycle from 1-11-12-13-14, prior to arriving at the optimal policy. This suboptimal policy has an average reward of 1. A discount rate, $\gamma=0.9$, is adopted for both the Q-learning and the Q_M -learning. The discount rate in this MDP has to be larger than 0.7579 such that the policy of servicing the mailroom is being chosen. Any discount rate less than 0.7579 will cause a larger value at state 11 than the value at state 2. Thus, the policy servicing the Printer room will look more attractive than the policy servicing the Mailroom. R-learning arrived at the optimal solution in about 3700 number of steps, whereas Q_M -learning and Q-learning took 10,700 and 13,500 steps, respectively, to arrive at the optimal

Fig. 3. Comparison of R-learning, Q-learning and Q_M -learning.

Fig. 4. Comparison of R-learning, Q-learning and Q_M -learning with eligibility traces.

solution. Comparing with R-learning in the undiscounted formulation, the learning with discounted formulation is prone to convergence to the suboptimal solutions. However, the maximum reward Q_M -learning performs slightly better than the cumulative reward Q-learning. The reason pertaining to this phenomenon will be discussed in the next simulation.

The reason that R-learning outperforming Q_M -learning in the previous simulation is investigated in Fig. 4. Eligibility traces formulations (Sutton, 1984; Sutton & Barto 1998) are introduced to the Q-learning and Q_M -learning with a large trace-decay parameter, $\lambda=0.9$. This ensures that the future rewards are apparent to the learning agent as soon as they are being discovered by a non-greedy exploratory policy. Fig. 4 shows that the Q_M -learning with eligibility trace has superior performance to the others. The Q_M -learning is more far-sighted than the others, by adopting the eligibility trace. Q_M -learning converges in about 1500 number of steps, while Q-learning and R-learning converges in about 3700 number of steps. The Q_M -learning converges to the optimal policy in the shortest possible timeframe without being dragged down by the suboptimal policy. This is consistent with the behaviour of maximum reward agent introduced in Section 2.2. The agent will attempt to discover the shortest path that lead to the highest reward, without concerning too much on the intermediate lower rewards. However, for cumulative rewards models such as Q-learning or R-learning, it takes time for them to fade away the influence of the lower reward and the effect of the suboptimal policy. Therefore, the convergence of these cumulative rewards models is not as immediate as the non-cumulative maximum reward model.

4. Optimal stopping problem in financial derivative pricing

In the original paper on average reward reinforcement learning (Schwartz, 1993), the author claimed that one of the reasons for discounting rewards is to compensate for the interest rate of future rewards, especially in the field of economic. This has been demonstrated by (Tsitsiklis & Van Roy, 1999; Van Roy, 1998) in the studies of an optimal stopping problem in financial derivative pricing. However, the

problem cannot be readily resolved by the discounted accumulative reward reinforcement-learning model. This is because the objective of the problem is to determine the optimal stopping state with the highest discounted non-cumulative rewards. This section attempts to demonstrate the application of the maximum reward reinforcement learning to model this financial derivative pricing problem.

4.1. Problem formulation

The optimal stopping problem was originally formulated by (Van Roy, 1998; Tsitsiklis, 1999) for optimal pricing prediction of an exotic derivative product (Parsley, 1997). The problem domain is a contract based on the daily prices of a single stock. When the contract is exercised, a payoff is received in an amount equal to the current price of the stock divided by the price a 100 days before. Please refer to (Tsitsiklis, 1999) on the background studies and details of this problem. The summary of this problem is given as follows.

The process of stock prices is generated based on a *geometric Brownian motion* model (Hull, 1997). The change in the stock price is modelled as a normal distribution with *drift rate* as the mean, and with *volatility* as the standard deviation, as described in Eq. (4.1).

$$\frac{\Delta p_t}{p_t} \sim \phi(\mu \Delta t, \sigma \sqrt{\Delta t}) \quad (4.1)$$

where

p_t is the stock price at time t ,

μ is the expected drift rate (percentage return, or interest rate of the stock price),

σ is the volatility (standard deviation of the stock price),

Δt is the interval of time, and

$\phi(m, s)$ is a normal distribution with mean m and standard deviation s

If the prices are computed on a daily basis, then the time interval is 1, and Eq. (4.1) is simplified into the update equation described in Eq. (4.2).

$$p_{t+1} = p_t(1 + \mu + \sigma\phi(0, 1)) \quad (4.2)$$

This problem considers the daily volatility of 0.02 and annual interest rate of 10%; the drift rate is 0.0004 by dividing the interest rate by the number of market opening days (~ 250 days). Therefore, the daily prices are updated based on Eq. (4.3).

$$p_{t+1} = p_t(1.0004 + 0.02\phi(0, 1)) \quad (4.3)$$

where $t \geq -100$.

The information available to the learning agent at a particular discrete time t is a vector, described as follows,

$$x_t = \left(\frac{p_{t-99}}{p_{t-100}}, \frac{p_{t-98}}{p_{t-100}}, \dots, \frac{p_t}{p_{t-100}} \right), \quad t \geq 0.$$

The payoff is rewarded if the contract is terminated at time t .

Therefore, the reward received a time step later is the last component of the information vector, $r_{t+1}=x_t(100)$. Then the value of the contract at time t is given by Eq. (4.4).

$$V_t = \sup_{\forall k \geq 0} E(\gamma^k r_{t+k+1}), \quad t \geq 0 \quad (4.4)$$

and the discount rate is given by, $\gamma=e^{-\mu}$. If t^* is an optimal stopping time, then the optimal value of the contract is described by Eq. (4.5).

$$V_{t^*} = E(r_{t^*+1}) = \sup_{\forall k \geq 0} E(\gamma^k r_{t^*+k+1}) \quad (4.5)$$

4.2. Learning model formulation and results

The state of the problem is constructed based on the following rationale. If there was a large slump and recovery in the stock prices within the past 100 days, then probably the contract should be held for some time to reap potential future benefits (Tsitsiklis, 1999; Van Roy, 1998). Therefore, the most important information is dependant on the current stock price, and the minimum stock price over the past 100 days. With this rationale, the state information is constructed as shown in Eq. (4.6). The first state dimension represents the proportional increment of the current stock price. The second state dimension implies the difference between the proportional increment of the current stock price over the minimum stock price of the past 100 days.

$$s_t = \left(\frac{p_t}{p_{t-100}}, \frac{p_t}{\min_{(t-100) \leq \tau \leq t} p_\tau} - \frac{p_t}{p_{t-100}} \right), \quad s_t \in \mathbb{R}^2 \quad (4.6)$$

In simulation trials, since the number of training iterations is always finite, there exist a maximum value in the set of the expected discounted payoff, and this maximum value is also the supremum of the set (Eq. (4.7)).

$$V_t = \max_{\forall k \geq 0} E(\gamma^k r_{t+k+1}) = \sup_{\forall k \geq 0} E(\gamma^k r_{t+k+1}), \quad (4.7)$$

$t \geq 0, \quad k$ is finite

Eq. (4.7) is equivalent to Eq. (2.6), which is the value function of the maximum reward reinforcement learning. Thus, this optimal stopping problem can be approximated with the maximum reward reinforcement learning, TD_M-learning, where the update error formula is shown in Eq. (2.15).

In this simulation, the optimal value function is approximated with the FITSK-RL model. The FITSK-RL model consists of a zero-order FITSK as the function approximator. The parameters for the learning are shown in Table 1.

The first and second dimensions are denoted as $s(1)$ and $s(2)$, which are computed using Eq. (4.6). The input ranges for the two state dimensions are $s(1) \in (0.5, 2.0)$ and $s(2) \in (0.0, 0.6)$, respectively. The number of pre-assigned membership

Fig. 5. Approximated optimal value function with maximum reward FITSK-RL.

functions for the two dimensions are 5 and 3. The total number of parameters to tune are $5 \times 3 = 15$. All the initial tunable parameters are set to zeros. The learning is iterated for 500,000,000 times. The approximated value function from the output of the model is plotted in Fig. 5.

Fig. 5 shows the surface plot of the approximated optimal value function with the proposed maximum reward FITSK-RL model after the learning iterations. The two state dimensions are represented with the X–Y axes, the approximated optimal value function is represented with the Z-axis. When the current stock price is near to the bottom of the past 100 prices ($s(2) \rightarrow 0$), then there is not much chance of getting better further benefits, and therefore expecting lower payoff. When the current stock price yields a growth of about 1.2 in $s(1)$, the expectation of payoff increases proportionally with $s(2)$. Increased in $s(2)$ means that there are larger slump and recovery over the past 100 days, and probably one should wait for sometime to reap potential higher payoff. The region at the upper right corner ($s(1) \rightarrow 2.0, s(2) \rightarrow 0.6$) are rarely encountered (see Fig. 5), and thus their learnt optimal value are lower than expected due to the lack of training examples. However, when such situation occurs, the process is being terminated immediately to reap the high benefits from these states. This is because the chance of getting any potential higher payoff than the current is quite slim.

Denoting the approximated optimal value function from the FITSK-RL as $\tilde{V}(s_t)$, then the stopping time, $\tilde{\tau}$, is computed by Eq. (4.8). This is similar to the terminating criterion defined by (Tsitsiklis, 1999), which was proven to be the optimal stopping time.

$$\tilde{\tau} = \min\{t | r_{t+1} \geq \tilde{V}(s_t)\} \quad (4.8)$$

The expected reward corresponding to the approximated optimal value function is estimated by averaging the rewards gathered over 50,000 independent trajectories, and terminated according to Eq. (4.8). The results are tabulated in Table 2. The expected reward computed from such stopping criterion is 1.289127, and the estimated standard deviation for this sample

Table 1
Predefined FITSK-RL parameters

Learning constant, $\alpha=0.001$	Discount rate parameter, $\gamma=e^{-\mu}=e^{-0.0004}$
Dimensions=2	Trace-decay parameter, $\lambda=0.5$

Table 2
Results of the optimal stopping problem

Strategy	Expected reward	Standard deviation	State dimension
Thresholding	1.238	–	–
Tsitsiklis model	1.282	0.0022	100
FITSK-RL	1.289	0.0029	2

mean is 0.002856. The expected reward is 1.238 from a thresholding strategy; and 1.282 (with 0.0022 standard deviation for sample mean) for the approximated optimal value function from (Tsitsiklis, 1999) formulation. Therefore, the expected payoff from the maximum reward FITSK-RL model is slightly better than the formulation by Tsitsiklis, which is leading by 0.007127. The expected reward is numerically 0.051127 better than the thresholding strategy. This means that the optimal pricing of the strategy identified by the maximum reward FITSK-RL model is 5% better than the thresholding strategy, and is about 1% better than the Tsitsiklis formulation.

4.3. Discussion

In the original paper on average reward reinforcement learning for the undiscounted reward problem, the author claimed that using discounting of rewards to compensate for the interest rate is unjustifiable (Schwartz, 1993). However, the optimal stopping problem in financial derivative pricing studied in this paper shows the contrary. The rewards are discounted by $e^{-\mu}$ to compensate for the possible future increment of the contract values, such that future values are comparable side-by-side with the current values. On the other hand, discounted cumulative reward reinforcement learning is not applicable to this problem. This is because the objective of the problem is to search for a highest expected reward for each state. Having this, a non-cumulative, discounted, maximum reward reinforcement learning formulation is shown to be directly applicable in solving this problem.

The FITSK-RL model is demonstrated to incorporate the maximum reward reinforcement learning formulation with ease in Section 2.3. The problem in this section is simulated with the maximum reward FITSK-RL model. The formulation presented in this section is slightly different to the one proposed in (Tsitsiklis, 1999). They formulated the problem with 100 state dimensions to demonstrate the good scaling of their local basis function approximator. However, the formulation in this paper reduces the state dimension to 2, such that the learning result can be easily visualized and inspected (as shown in Fig. 5). This improves the interpretability of the learning result. Furthermore, the operations of FITSK-RL model are systematically defined, as opposed to the manually generated local basis function approximation method in (Tsitsiklis, 1999). The expected payoff of the maximum reward FITSK-RL model with 2 state dimensions is slightly better than the existing methods with 100 state dimensions. This shows that the state definition in Eq. (4.6) with two dimensions can adequately provide enough information to the learning system. This is a

significant improvement in modelling complexity over the existing methods.

5. Conclusion

In this paper, a new maximum reward criterion is proposed as the objective of reinforcement learning. The maximum reward criterion is subsequently incorporated into the reinforcement learning update formulae from the Markov decision process perspective. Subsequently, the maximum reward reinforcement learning is mapped to the generic reinforcement-learning framework to solve the modelling issue of this new learning paradigm.

The distinct characteristic of the maximum reward reinforcement learning is a non-cumulative, maximum reward-oriented behaviour of the learning agent, as opposed to the usual cumulative rewards-oriented behaviour. It attempts to search for the policy that leads to the largest reward, regardless of intermediate lower rewards. This positions it to have superior immunity to the existence of sub-optimal policies. This is demonstrated with the example in Section 3, where the maximum reward agent established a convergence to the optimal policy in the shortest timeframe. However, other agents with underlying cumulative rewards-oriented behaviour are being dragged down by the sub-optimal policy and lower reward. As a result, they suffer from slower convergence to the optimal policy.

Section 4 demonstrates a successful application of maximum reward reinforcement learning to asymptotically approximate the supremum of payoff values in an optimal stopping problem. The maximum reward FITSK-RL model proposed in Section 2.3 has been adopted to resolve this problem. This problem cannot be effectively solved using the usual cumulative rewards reinforcement-learning paradigm because the underlying payoff values are non-cumulative. Despite reducing the problem to a two-state dimension problem, the approximation result using the FITSK-RL model is slightly superior to the original formulation with 100 state dimensions in (Tsitsiklis & Van Roy, 1999). Furthermore, this improves the interpretability of the learning results and reduces the modelling complexity of the problem. The approximation result is visualized such that the rationale of the result can be examined with ease. This would not be possible with the original formulation of the problem as the state dimensions are too large and it is hard to gain insight into the learning result of the manually defined local basis functions. However, this is possible with the FITSK-RL model as the operations are systematically defined, and the state dimensions are small.

In summary, the maximum reward reinforcement learning approach is proposed as a possible new learning paradigm to exhibit superior performance for problems with underlying non-cumulative nature of reward values. Examples and simulations are provided to illustrate this. The modeling issue of the maximum reward reinforcement learning is solved by mapping it into the generic reinforcement learning framework. However, the theoretical research of maximum reward reinforcement learning is still at its infancy stage, particularly from the optimal control perspective. The convergence of maximum reward reinforcement learning

resulted from a linear function approximator is yet to be established. Some possible resemblance of the convergence result may possibly be extended from the Tsitsiklis' papers (Tsitsiklis & Van Roy, 1997, 1999), which justifies a need of possible further research. The relationship to the sensitive discount optimality criterion (Mahadevan, 1996; Puterman, 1994) has not been studied, and is worthy of a further research effort.

References

- Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.
- Hull, J. C. (1997). *Options, futures, and other derivatives* (3rd ed.). Englewood Cliffs, NJ: Prentice Hall.
- Kaelbling, L. P., Littman, L. M., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237–285.
- Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22, 159–196.
- Parsley, M. (1997). Exotics enter the mainstream. *Euromoney*, 127–130.
- Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York, NY: Wiley.
- Quah, K. H., & Quek, C. (2006). FITSK: Online local learning with generic fuzzy input Takagi-Sugeno-Kang fuzzy framework for nonlinear system estimation. *IEEE Transactions Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1), 166–178.
- Quah, K. H., & Quek, C. (submitted for publication). FITSK-RL: Function approximation with the fuzzy input Takagi-Sugeno-Kang model for a generic reinforcement learning framework.
- Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. *Technical report CUED/F-INFENG/TR166*, Cambridge University Engineering Dept. http://svr-www.eng.cam.ac.uk/reports/svr-ftp/rummery_tr166.ps.Z.
- Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards, *ICML 1993*, pp. 298–305.
- Sutton, R. S., (1984). *Temporal credit assignment in reinforcement learning*, PhD dissertation, University of Massachusetts, Amherst, MA.
- Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3(1), 9–44.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: The MIT Press.
- Tadepalli, P., & Ok, D.-K. (1998). Model-based average reward reinforcement learning. *Artificial Intelligence*, 100, 177–224.
- Tsitsiklis, J. N., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5), 674–690.
- Tsitsiklis, J. N., & Van Roy, B. (1999). Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 44(10), 1840–1851.
- Van Roy, B. (1998). *Learning and value function approximation in complex decision processes*, PhD dissertation, Massachusetts Institute of Technology.
- Watkins, C. J. C. H. (1989). *Learning with delayed rewards*, PhD dissertation, Psychology Department, Cambridge University, England.